Baden-Wuerttemberg Cooperative State University
Mannheim

Bachelor Thesis

Conception and Development of a Machine Learning Model for the Analysis of the Energy Consumption of a Plasterboard Dryer

Business Informatics

Software Engineering

Author:	Marius Memmel
Matriculation Number:	2225721
Company:	Knauf Information Services GmbH
Department:	Process Optimization & Energy Management
Course:	WWI16SEA
Course Director:	Prof. Dr. Julian Reichwald
Academic Advisor:	Prof. Dr. Julian Reichwald julian.reichwald@dhbw-mannheim.de +49 (0)621 4105 - 1395
Company Advisor:	Dr. Halgurt Bapierre halgurt.bapierre@knauf.de +49 9323 31 - 5418
Project Period:	24.05.2019 – 16.08.2019

Abstract

Title	Conception and Development of a Machine Learning Model for the Analysis of the Energy Consumption of a Plasterboard Dryer
Author:	Marius Memmel
Course:	WWI16SEA
Company:	Knauf Information Services GmbH

This bachelor thesis deals with the analysis of a plasterboard dryer's energy consumption and a possible machine learning approach to produce an accurate model that later can be used for energy optimization.

It is of high interest for the Knauf group to identify the factors influencing the energy consumption of plasterboard driers, which their plants use to remove moisture from the gypsum plasterboards in the production process. These driers consume most of the energy needed to run a plant. By utilizing the Cross-Industry Standard Process for Data Mining (CRISP-DM), the business partners want to gain a deeper insight into the influencing factors. The ResMa® Energy and Resource Manager (ResMa) supports this process by providing sensor data of the plant. The first step of the process features a conversation with the business experts to identify the business objectives and to gather valuable domain knowledge regarding the problem. Following the data export, the data understanding phase utilizes statistical and visualization techniques to analyze the data. The findings collected build the foundation of preparing the data for the modeling step. This step uses the prepared data to train different machine learning models. Different metrics help to evaluate the models against each other and to rank them accordingly. After evaluation, the models are in the center of discussion with the business to validate the objective fulfillment and to repeat the process if necessary. A successful cycle results in a model that fits the business objectives and can enter the deployment phase.

An academic explanation of CRISP-DM and common machine learning practices and algorithms promote an understanding of the project's foundations. Acting as a guideline, the CRISP-DM is the basis of a concept development. Part of this concept are the analysis of the actual state of the Knauf group and the formulation of a hypothesis. The proof of concept later implements the developed concept. An evaluation and a discussion of the implementation finally lead to acceptance or rejection of the hypothesis. An evaluation of the project work summarizes the results and an outlook presents a glimpse into future projects.

Contents

List of Figures

List of Tables

List of Model Parameters

List of Acronyms

ANN Artificial Neural Network

CRISP-DM Cross-Industry Standard Process for Data Mining

DNN Deep Neural Network

EDA Exploratory Data Analysis

MAE Mean Absolute Error

MSE Mean Squared Error

NFLT No Free Lunch Theorem

R^2 R Squared

ResMa ResMa® Energy and Resource Manager

RMSE Root Mean Squared Error

SGD Stochastic Gradient Descent

SVM Support Vector Machine

SVR Support Vector Regression

1 Introduction

In the year 1932, the brothers Alfons and Karl Knauf marked the beginning of the Knauf Group by securing the mining rights to gypsum deposits in Schengen, Obermosel.[1] In 1933 the first gypsum factory was established by the two brothers in Perl at the Moselle.[1] After the relocation to Iphofen, Bavaria in 1949, Knauf created a completely new construction method for the production of fully new gypsum boards.[1] Until today these have been the foundation of a big variety of gypsum-based products offered by the company. Like Dr. Hans Wintrich, leader of the process optimization and energy management in the Knauf group showed in his presentation at the 2nd energy dialog Main-Franconia in the year 2014, the cost for electricity in Germany is rising rapidly.[cf. 2, p. 5] But not only the cost factor drives the company to reduce energy consumption, but also its commitment to people, nature and society. He points out that Knauf achieved improvements of up to 10% before by modifying its machinery.

With machine learning and big data becoming very popular over the past decade, Knauf has been offered the opportunity to react to the issues mentioned above by gaining deep insights to the influential factors and therefore being able to optimize in more detail. The company operates more than 220 plants all over the world. Most of those are built in a similar manner, which makes optimization scale on a global basis. By reducing the energy consumption of a component that is used several times, the environmental footprint and the cost of energy can be reduced drastically. One of these components is the dryer used to remove moisture from the plasterboards. Because it is one of the most energy-consuming parts of the plant and is operated in plants all over the world, it is a perfect candidate to attempt such energy optimization.

The environment and the dryer itself are equipped with several sensors, measuring data and storing it in the ResMa® Energy and Resource Manager (ResMa). This means that enough data is expected to be available for analysis and to apply machine learning algorithms. This thesis focuses on performing CRISP-DM to gather insights that are of value for the operating business. It also includes the training of machine learning models to depict reality and to provide a starting point for optimization.

2 Theoretical Foundations

2.1 Cross-Industry Standard Process for Data Mining

2.1.1 Business Understanding

When carrying out data mining projects, "[the] goal of CRISP-DM is to make large data mining projects faster, cheaper, more reliable, and more manageable."[3, p. 457] "The process model is independent of both the industry sector and the technology used."[cf. 4, abstract] These attributes make the process highly flexible and fit for most applications. Furthermore, the process is divided into six phases that are visualized in 2.1

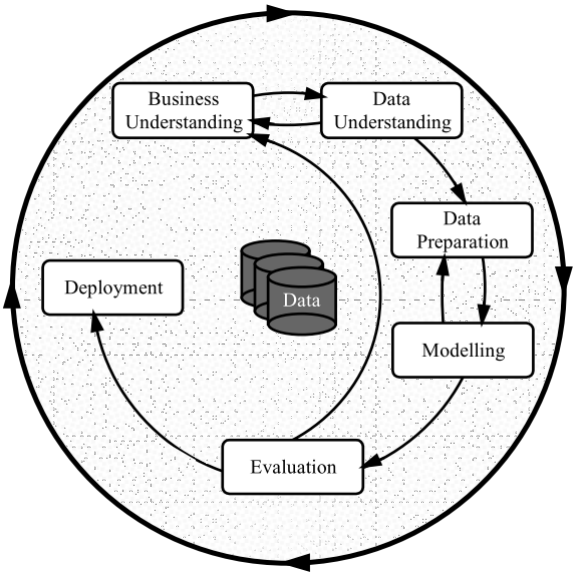


Figure 2.1
The CRISP-DM Cycle - Source: [4]

In the first phase, one should try to understand the project objectives and requirements focusing on the business point of view by assessing the situation, assumptions, and constraints.[cf. 4, p. 5f. cf. 3, p. 457] Also unknown but import factors that the business might not be aware of but might affect the process must be uncovered and taken into consideration.[cf. 3, p. 457] It is then that the obtained knowledge about the business needs is translated to data mining goals.[cf. 4, p. 5f. cf. 3, p. 458]

Furthermore, the type of data mining problem has to be specified, e.g. classification or regression.[cf. 3, p. 458] Depending on this decision, criteria for model assessment have to be defined including performance metrics and benchmarks.[cf. 3, p. 458]

As the result of this phase, a project plan should be designed including these assumptions and constraints to achieve and measure the defined objectives.[cf. 3, p. 457] The business understanding is seen as the core step of a successful data mining project.[cf. 5, p. 191]

2.1.2 Data Understanding

In the next phase, data is collected and analyzed.[cf. 4, p. 5f.] To identify what data to collect, a list of the necessary data or databases and their data types should be created.[cf. 3, p. 459] The process of obtaining the data should be documented and used tools should be listed as well as occurring problems.[cf. 3, p. 459] [Steinwart & Christman 3] also suggest describing the main properties of the data using a *data description report.*[cf. 3, p. 458] This report includes "quantity of data [...], format of the data, coding, percentage and patterns of missing values, identifier variables needed for merging data from different databases or tables [and the] time period when the data was collected."[3, p. 459]

After the metadata was described, an *exploration report* should be created, describing the distribution of the key attributes.[cf. 3, p. 459] Categorical attributes can be analyzed using a list of possible values and a contingency table displaying connections between those.[cf. 3, p. 459] If the values are continuous, descriptive statistics like "[...] minimum and maximum values, mean and standard deviation, or their robust pendants, such as median and median absolute deviation [...]" must be used instead.[3, p. 459]

By describing and exploring the data, first insights can be gained and hypothesis about the connections can be formed.[cf. 4, p. 5f.] It is also important to verify the data quality because poor quality might not live up to the expectations of the project and lead to insufficient models later on.[cf. 4, p. 5f.] If a huge dataset is collected, a reduction can be considered to reduce complexity.[cf. 5, p. 192]

2.1.3 Data Preparation

The data preparation phase describes the transformation of the raw data to the final dataset that will be fed into the algorithms, resulting in a model. [cf. 4, p. 5f.] [Wirth & Hipp 4] notes that this process has to be repeated several times in no particular order until the data is cleaned up sufficiently.

First, a selection of attributes is made using assumptions and insights from the preceding steps. Then the data is cleaned up by "[correcting] typing errors [,] the transformation of existing attributes [...] and by defining derived attributes."[3, p. 460] Derived attributes can lead to dimensionality reduction if two or more attributes are combined to a single metric.[cf. 3, p. 460]

2.1.4 Modeling

In the modeling phase, various modeling techniques are selected and applied to the dataset which was prepared in the preceding phase.[cf. 4, p. 5f.] The selection is influenced by the data mining goals, the properties of the data and the assumptions made.[cf. 3, p. 461] Most modeling techniques have parameters called hyperparameters that can be calibrated, leading to higher performance.[cf. 4, p. 5f.] It also has to be paid attention to the data format each specific algorithm requires or can deal with best.[cf. 4, p. 5f.]

In the next step, a test design is created to measure the accuracy or precision of the model's predictions [in case of a classification task] and to make sure the dataset was not overfitted.[cf. 3, p. 464] "A common practice in data mining projects is to split the dataset randomly into three parts called the training dataset, validation dataset, and test dataset."[3, p. 464] The training set is used to train several models which are tested using the test set.[cf. 3, p. 465] To calibrate the model's parameters the validation set can be used to avoid overfitting the test set.[cf. 3, p. 465]

After preparing and splitting the data, the selected algorithms are trained using the training set.[cf. 3, p. 465f.] Furthermore, the validation set is used to identify the best hyperparameters for each model and the model's performance is measured using the test set.[cf. 3, p. 465f.] The fitted models are then described by their hyperparameters, predictions, and performance on the test set.[cf. 3, p. 465f.]

The model building step ends with the model assessment.[cf. 3, p. 466] This includes an interpretation of the model taking into account the defined goals.[cf. 3, p. 466] Ranking the models by their evaluation criteria can also be helpful in this process.[cf. 3, p. 466] This step combined with the model building step forms an iterative process that can improve the model's hyperparameters and should be carried out as long as improvement can be made by tuning the parameters.[cf. 3, p. 466] One can move freely between the modeling and the data preparation phase because problems might arise in the modeling process concluding in changes must be made on the data.[cf. 4, p. 5f.]

2.1.5 Evaluation

In this phase, an evaluation of the results is made by discussing the findings with the business analysts and domain experts to ensure that not only the technical success criteria but also that the business success criteria are fulfilled.[cf. 3, p. 466] It is there the business confirms whether or not the model is sufficient for their use case and if the budget and time constraints are met.[cf. 3, p. 466]

"The review process summarizes the whole data mining process so far and describes the unexpected findings, missed activities, and overlooked potential risk factors and lists actions that should be repeated."[3, p. 466] Depending on the outcome of the evaluation it is determined what steps are taken next and if a transition to the deployment phase can be made.[cf. 4, p. 5f.]

2.1.6 Deployment

The last phase focuses on the deployment of the models produced and the insights gathered.[cf. 4, p. 5f.] Depending on the customer, the knowledge has to be presented in the form of a report, an application or even an "implementation of a repeatable data mining process [...]".[4, p. 5f.] Additionally plans have to be made as to how the deployment is monitored and maintained in the future.[cf. 4, p. 6]

At the end of the process, a final report has to be produced summing up the procedure and findings of the project.[cf. 3, p. 466] Also in respect to future data mining projects, a project review can be used to document the usefulness of different approaches, pitfalls that one should avoid, and recommendations or problems when using certain software tools.[cf. 3, p. 466]

2.2 Machine Learning Terminology

2.2.1 Definition

> "[Machine learning is the] field of study that gives computers the ability to learn without being explicitly programmed." - Arthur Samuel, 1959 [6, p. 4]

Arthur Samuel first coined the term machine learning in 1959, with this definition. Focusing on machine learning as a field of study, a younger definition was coined by [Harrington 7], stating that it is a discipline "[lying] at the intersection of computer

science, engineering, and statistics and often appears in other disciplines [...]" corresponds to his idea.[7, p. 5] It goes against the common belief that a computer has to be explicitly programmed in order to learn how to perform a certain task.

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." – [8, p. 2]

[Mitchell 8] presents a more "engineering-oriented" approach. To make his definition easier to understand, most textbooks use the example of spam emails. The program is said to be a spam filter using machine learning to classify whether an email is spam or not. To learn how to do so, a training set consisting of examples of both spam emails and regular emails is used. "Each training example is called a training instance (or sample)."[6, p. 4] The task T for the program is to flag incoming emails as spam or to forward regular ones. Experience E is the training data from which the algorithm can learn. Furthermore, a performance measure P has to be defined. [Geron 6] suggests using the ratio of correctly classified emails as the performance measure. In the context of machine learning, "[this] particular performance measure is called accuracy and is often used in classification tasks."[6, p. 4]

> "Machine learning, then, is about making computers modify or adapt their actions [...] so that these actions get more accurate, where accuracy is measured by how well the chosen actions reflect the correct ones." – [9, p. 4]

[Marsland 9] focuses on a computer performing actions and improving the accuracy of those, narrowing the role of the computer down to an actor like instance. A computer then has the ability to modify and adapt those actions with the goal of increasing the performance. Like [Geron 6] in the example before, he is suggesting using accuracy to measure performance. It can be noticed that his definition of accuracy is formulated more broadly and is not limited to classification tasks, but on the actions reflecting the correct ones.

Machine learning can be divided into four types of learning, each different depending on the type of task to perform. The types are further described in the following.

2.2.2 Supervised Learning

Supervised learning is a form of learning in which the training set consists of input data already including the desired target data. The target represents the output that the algorithm should produce.[cf. 9, p. 6, cf. 7, p. 178] The target or target value in the training set is also called label.[cf. 6, p. 10] If every possible piece of input data is

provided, the approach lies near to just use the data as a look-up table without machine learning being involved.[cf. 9, p. 6] The reason why machine learning is preferred over this solution is that it can generalize well on new data that the algorithm has not seen before.[cf. 9, p. 6] By doing so, it can deal with noise, small inaccuracies in the data, that is inevitable when dealing with real-world data.[cf. 9, p. 6] According to [Harrington 7], supervised learning is further split into classification and regression, depending on the type of the desired output.

Classification aims to use a classifier built by a machine learning algorithm to predict new data into classes.[cf. 10, p. 59, cf. 7, p. 10] The example of a spam-filter mentioned above is a classification task because the training set already includes whether the email is spam or not. Another application is to classify images of handwritten digits into classes from 0 to 9.[cf. 10, p. 59]

Regression differs from classification by its output value. The "[...] target variable is numeric and continuous."[7, p. 10] Examples include the prediction of dollar values in a financial model and a prediction of the price of a car given a set of features.[cf. 10, p. 68, cf. 6, p. 10] "Note that some regression algorithms can be used for classification as well, and vice versa. For example, Linear Regression is commonly used for classification, as it can output a value that corresponds to the probability of belonging to a given class (e.g., 20% chance of being spam)."[6, p. 10]

2.2.3 Unsupervised Learning

Unsupervised learning is the opposite of supervised learning. In this case, the desired target or label of the training data is not provided.[cf. 7, p. 10, cf. 6, p. 10] This means that the algorithm has to come up with outputs without the possibility of verifying those by given data.

A selection of tasks that are performed using unsupervised learning are:

- Clustering
- Dimensionality Reduction
- Anomaly Detection

Clustering tries to group similar items.[cf. 7, p. 10] The training data is fed to the algorithm, which then tries to cluster the given data into a several groups.[cf. 6, p. 10] The algorithm has no information about what groups might exist or how the instances are connected[cf. 6, p. 10] To subdivide the identified groups, a hierarchical clustering algorithm can be used.[cf. 6, p. 10]

Dimensionality reduction reduces the number of features or dimensions to make it possible to visualize, speed up the algorithm, reduce disk and memory space and

sometimes even improve model performance.[cf. 7, p. 10, cf. 6, p. 12] To visualize the dimensions properly, a number of two or three must be reached.[cf. 7, p. 10] The dimensions are equivalent to the number of attributes. By e.g. merging several correlated features into one, the dimension can be reduced by the number of eliminated attributes.[cf. 6, p. 12]

To identify unusual data points, *anomaly detection* can be performed. By training such an algorithm with normal instances, it can detect whether a new instance is normal or unusual.[cf. 6, p. 12] In machine learning this can be used to detect and remove outliers before feeding it into another machine learning algorithm.[cf. 6, p. 12]

2.2.4 Semi-supervised Learning

Semi-supervised learning combines supervised and unsupervised learning. It can deal with partially labeled data.[cf. 6, p. 14] Usually a larger amount of unlabeled data is used, supported by a small amount of labeled data.[cf. 6, p. 13]

A common use case is identifying persons or objects on unlabeled images.[cf. 6, p. 13] First, a label is created, identifying an object on a picture.[cf. 6, p. 14] This label is then used to identify the same object on new images without the need for a manual labeling process.[cf. 6, p. 13]

2.2.5 Reinforcement Learning

Reinforcement learning is a type of learning in which a "[...] learning system, called an agent in this context, can observe the environment, select and perform actions, and get rewards in return [...]".[6, p. 13f.] Those rewards can also be negative, this is when they are called penalties.[cf. 6, p. 13f.] The agent must explore the environment using different actions.[cf. 9, p.6]

The goal is to figure out a strategy, or policy, to maximize the reward over time.[cf. 6, p. 13f.] Depending on this policy, the actions are determined that the agent performs in the given situation.[cf. 6, p. 13f.] It is important to mention, that the algorithm is only told when its decision was wrong, but not how to correct it.[cf. 9, p.6] The decision process is only reinforced by rewards.[cf. 9, p.6]

2.3 Machine Learning Models

2.3.1 No Free Lunch Theorem

When it comes to the selection of the algorithm, the No Free Lunch Theorem (NFLT) by [Wolpert 11] states that when dealing with a problem, no prior assumption can be made what kind of algorithm would be the ideal solution.[cf. 11, cf. 12, p. 163] In the context of machine learning this means that without making any assumptions about the dataset, no algorithm can be declared superior to others.[cf. 6, p. 30f.] To know for sure which algorithm performs best on the given problem, all algorithms have to be evaluated.[cf. 6, p. 30f.] To speed up this process, assumptions about the data can be made to narrow down possible solutions.[cf. 6, p. 30f.] For example distinctions about the learning type and the learning task can be made to find a subset of possible algorithms.

In summary, if an algorithm is to be selected, initial assumptions about the data must be made to narrow down possible solutions. Multiple algorithms must be implemented, and the performance of the selected algorithms must be compared. To narrow down the number of models to discuss, only supervised regression algorithms are explained in the following. Classification is not covered and if so, only to support the explanation of a regression algorithm.

2.3.2 Linear Regression

Mathematical Foundations

As the name suggests, linear regression uses a linear model to represent the data and to make predictions. The input data for this type of regression is a matrix X, containing the features and the predictions represented by vector $\boldsymbol{y}$.[cf. 7, p. 155] Note that outside of the equations, matrices will be written in capital letters, vectors will be written in bold lower case letters and variables will be written in lower case letters. To predict e.g. y_1, so-called weights have to be applied to $\boldsymbol{x_1}$, a row of matrix X. The regression weights are represented by vector $\boldsymbol{w}$. Because at the beginning the weights do not fit the training data, the predictions are expected to be of high error. To reduce this error and improve the predictions, the weights have to be updated by learning from the training data. The process of fitting the model to the data is described in the following based on [Géron 6], [Patterson 12] and [Harrington 7].

Linear regression produces an output or prediction by computing the weighted sum of the input features.[cf. 6, p. 106] Additionally, a constant is added called the bias term.[cf. 6, p. 106] The resulting equation is shown in 2.1. Each input feature x_i is

multiplied by its corresponding weight w_i. Those multiplications are summed up and the bias term w_0 is added to produce the output $\hat{y}$ for input $\boldsymbol{x}$. Note that x_0 is set 1 to not influence the bias.[cf. 6, p. 107]

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n \tag{2.1}$$

$\hat{y}$ predicted value
n number of features
x_i i^{th} feature value
w_j j^{th} model parameter (w_0 is the bias term, w_1,w_2, ..., w_n are the feature weights)

2.1 can be written in the vectorized form to simplify the formula and to make future calculations easier. Because $\boldsymbol{x}$ and $\boldsymbol{w}$ both have the dimension $(m, 1)$, they can not be multiplied. This can be solved by transposing $\boldsymbol{x}$, changing its dimensions from $(m, 1)$ to $(1, m)$. Multiplying $\boldsymbol{x}^T$ by $\boldsymbol{w}$ then results in a $(1, 1)$ matrix, being a single value representing $\hat{y}$, the predicted result.

$$\hat{y} = x^T w \tag{2.2}$$

x instances feature vector, x_0 to x_n with x_0 always equal to 1
w model's parameter vector, bias term w_0 and feature weights w_1 to w_n
x^T transpose of the x vector
$\hat{y}$ predicted value

The final model consists of the weights $\boldsymbol{w}$ that can be applied to its corresponding features $\boldsymbol{x}$. Due to the linear form of the equation, a single feature can be visualized. In this context, the model tries to find the best fitting line for the dataset and uses this line to predict the $\hat{y}$ for new inputs $\boldsymbol{x}$. An example of visualizing linear regression is shown in 2.2. Visualizing multiple features becomes more difficult and finally gets impossible, as the simple line is replaced by multidimensional hyperplanes.

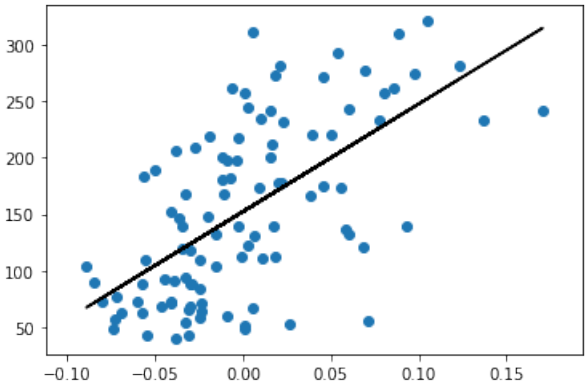


Figure 2.2: Linear Regression Visualized. w_0 is the y intercept and w_1 is the slope.

Normal Equation

To adapt the weights, it has to be measured how the weights contribute to the predicted value $\hat{y}$. To do so, an error is introduced to the equation.[cf. 7, p. 155] This error should be minimized to result in the best possible $\boldsymbol{w}$ which is named $\hat{\boldsymbol{w}}$.[cf. 7, p. 155] In this example, the Mean Squared Error (MSE) is used. First, the square of the differences between the expected and the predicted values are calculated. Then the results are summed and divided by the number of samples, resulting in the mean.

$$MSE(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \tag{2.3}$$

Because the differences are squared, negative values can not cancel out the positive values.[cf. 7, p. 155] Additionally, multiple input vectors $\boldsymbol{x}$ can be combined to a matrix X. To produce the output vector $\hat{\boldsymbol{y}}$, the weights then have to be applied to X instead.

$$MSE(\hat{y}_i, y_i) = \frac{1}{m} \sum_{i=1}^{m} \left(x_i^T w - y_i \right)^2 \quad \text{with} \quad \hat{y}_i = x_i^T w \tag{2.4}$$

$$MSE(\hat{y}, y) = \frac{1}{m} (Xw - y)^2 = \frac{1}{m} (Xw - y)^T (Xw - y) \quad \text{with} \quad \hat{y} = Xw \tag{2.5}$$

m number of samples
x_i x vector of the i^{th} sample
$\hat{y}_i$ y value of x_i
w model parameter vector
X X matrix (x vectors of all samples)
$\hat{y}$ $\hat{y}$ vector ($\hat{y}$ values for all samples, representing the predictions)
y y vector (y values for all samples, representing the predictions)

The MSE is a convex loss function for linear regression, so its minimum can be found by calculating the partial derivative with respect to $\boldsymbol{w}$ and set the resulting equation equal to 0.[cf. 12, p. 73]

$$\frac{\partial J(w)}{\partial w} = X^T (y - Xw) \tag{2.6}$$

$\frac{\partial J(w)}{\partial w}$ partial derivative of the loss function $J(w)$ with respect to w

X^T transpose of the X matrix

The equation then has to be solved by $\boldsymbol{w}$, results in a *closed-form solution* for $\hat{\boldsymbol{w}}$.[cf. 6, p. 108] This equation is called the *Normal Equation*.[cf. 6, p. 108] It has to be mentioned that for this equation to work, an inverse of the term $(X^T X)$ has to exist and should be checked before calculation.[cf. 7, p. 156]

$$\hat{w} = \left(X^T X \right)^{-1} X^T y \tag{2.7}$$

$\hat{w}$ w vector that best fits the samples
$(X^T X)^{-1}$ matrix inversion of $(X^T X)$

Gradient Descent

Another way of finding $\hat{\boldsymbol{w}}$ is by using an algorithm called *gradient descent*. This algorithm is often used in machine learning to minimize a loss function. A loss function represents the error distribution the weights produce on predictions. Instead of finding the optimal solution with one costly calculation like the normal equation, gradient

descent tries to tweak the weights iteratively until a minimum is reached.[cf. 6, p. 111] Gradient descent measures the local gradient or slope of the loss function with regards to the weight vector $\boldsymbol{w}$ by taking the derivative of the loss function.[cf. 6, p. 111, cf. 12, p. 30] It then takes a step in the direction of the descending gradient.[cf. 6, p. 111] Eventually, the gradient will become zero, signalizing that a minimum is reached.[cf. 6, p. 109] At the beginning of the process, the weights $\boldsymbol{w}$ are initialized randomly.[cf. 6, p. 111] This is called random initialization.[cf. 6, p. 111] Then they are improved gradually by taking small steps towards the minimum until the algorithm converges to the minimum.[cf. 6, p. 111] When using a convex loss function, as the MSE in linear regression, only one global minimum is present, leading the algorithm to the optimal solution.[cf. 12, p. 30, cf. 6, p. 113]

The size of the steps the algorithm takes can be influenced by the learning parameter. If the parameter is set too small, the algorithm will have to perform many iterations to converge.[cf. 6, p. 111] If the parameter is set too high, the algorithm will shoot over the minimum, possibly reaching an even higher position.[cf. 6, p. 112] This can lead to the algorithm diverging with higher values, failing to find a good solution.[cf. 6, p. 112]

Regularization

If the dataset consists of more features than samples, the "[...] data matrix X isn't full rank [...]" which means that an error will occur trying to compute the inversion in the normal equation. To tackle this issue and to also guard against overfitting, regularization can be applied to the linear regression model.[12, p. 164] *Ridge regression* is one of two shrinkage methods.[cf. 12, p. 164] Originally designed for cases in which more features than samples exist, it can also be used to add bias to the model, leading to better predictions on new data points.[cf. 12, p. 164] To regularize the model, a regularization term is added to the cost function used for training.[cf. 6, p. 127] The model then has to not only reduce the error but also take the regularization term into account.[cf. 6, p. 127] This leads to weights being pulled close to zero.[cf. 6, p. 127] The bias term is not regularized, which can be seen in the equation by the sum starting at index 1 instead of 0.[cf. 6, p. 127f.] The regularization term consists of $\frac{1}{2}$ the l_2 norm or *euclidean norm* of the weight vector multiplied by a hyperparameter α, controlling the amount of regularization applied.[cf. 6, p. 39, 127f., 130] If α is equal to 0, ridge regression is equal to linear regression.[cf. 6, p. 127f.] If α is very large, all weights are pulled very close to zero, resulting in a flat line going through the datasets mean.[cf. 6, p. 127f.] [Géron 6] mentions, that ridge regression is very sensitive to the scale of the input features, so the dataset should be scaled before training.

$$J(w) = MSE(w) + \alpha \frac{1}{2} \sum_{i=1}^{n} w_i^2 \tag{2.8}$$

Figure 2.3: MSE with l_2 Norm

A different method of regularizing linear regression is using *lasso regression*. Different to ridge regression, it uses the l_1 norm, sometimes called the *manhatten norm*. Again the bias is not regularized. Instead of pulling the weights close to zero, "[...] it tends to completely eliminate the weights of the least important features [...]".[6, p. 130] This leads to an automatic feature selection resulting in a sparse model with fewer weights.[cf. 6, p. 130]

$$J(w) = MSE(w) + \alpha \sum_{i=1}^{n} |w_i| \tag{2.9}$$

Figure 2.4: MSE with l_1 Norm

Elastic net combines ridge and lasso regression. The regularization term consists of both regularization terms with an additional hyperparameter r that controls the mix between the two. "When $r = 0$, elastic net is equivalent to ridge regression, and when $r = 1$, it is equivalent to lasso regression [...]".[6, p. 132] [Géron 6] suggests to always use "[...] at least a little bit of regularization [...]" and therefore not to use plain linear regression.[6, p. 132] In his opinion, ridge regression is a good default to use, but if it looks like only a few features are important, lasso regression or elastic net should be preferred to eliminate the unimportant features.[cf. 6, p. 132] As lasso regression tends to behave erratically when the number of features is greater than the number of training samples or "[...] when several features are strongly correlated [...]", elastic net should be used over lasso regression.[6, p. 132]

$$J(w) = MSE(w) + r\alpha \sum_{i=1}^{n} |w_i| + \frac{1-r}{2} \sum_{i=1}^{n} w_i^2 \tag{2.10}$$

Figure 2.5: MSE with l_1 and l_2 Norm

2.3.3 Polynomial Regression

If the data is too complex to be represented by a simple line, a variation of linear regression called polynomial regression can be used.[cf. 6, p. 121] This type of regression creates additional features by using the powers of features and combinations among

them.[cf. 6, p. 122f.] This enables the algorithm to find relationships between features which is what a linear regression sometimes lacks.[cf. 6, p. 122f.] An example of the resulting features for a *polynomial degree = 3* and a *number of input features = 2* is given in 2.11.[cf. 6, p. 123]

$$y = 1w_0 + w_1a + w_2b + w_3a^2 + w_4b^2 + w_5a^3 + w_6b^3 + w_7ab + w_8ab^2 + w_9ba^2 \quad (2.11)$$

a, b input features

It can be seen, that even at a low degree and a small number of input features, the resulting array containing the features has a size of *10*. The resulting size in relation to the number of input features n and the polynomial degree d can be calculated by 2.12.[cf. 6, p. 123]

$$\frac{(n+d)!}{d!n!} \quad (2.12)$$

n number of input features
d polynomial degree

By increasing the polynomial degree, the risk of overfitting the training data increases too. To prevent this from happening, the learning curve should be monitored continuously. The training error should always be as close as possible to the validation error, because otherwise, it would be a sign of overfitting the training set.

2.3.4 Support Vector Machines

A Support Vector Machine (SVM) can be used to perform many different tasks in machine learning, including linear or non-linear classification, regression and outlier detection.[cf. 6, p. 145] To understand the context, first the approach for classification is explained to later build the bridge to regression.

To separate the different classes in the dataset, the SVM tries to fit a line through the data. "[This] line not only separates the two classes but also stays as far away from the closest training instances as possible."[6, p. 145] This area should be as large as possible and should not contain any data points.[cf. 9, p. 170] The data points that the lines go through are called "support vectors"[cf. 9, p. 170]. The distance between these vectors and the line parallel to those is called margin M.[cf. 9, p. 170] It can be said that the margin should be as large as possible. The support vectors are the most

important points because no other points are used for the classification.[cf. 9, p. 170] The rest of the data can be discarded leading to savings in storage.[cf. 9, p. 170]

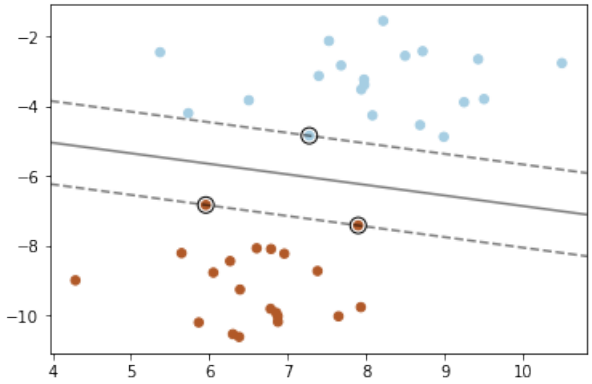


Figure 2.6: SVM Visualized. The circled points represent the support vectors. Source: [13]

To use this approach for a regression task, the objective has to be reversed.[cf. 6, p. 154] Instead of maximizing the margin M between the lines, as many points as possible should now be fit in this area.[cf. 6, p. 154] The margin therefore has to be restricted by a hyperparameter called ϵ.[cf. 6, p. 154] Predictions are then made using the line that was fit through the data and which is parallel to the two support vectors. An implementation of SVM performing a regression task is called Support Vector Regression (SVR). Because adding more data points within the area between the outer lines does not affect the predictions, the model is $\epsilon - insensitive$.[cf. 6, p. 154] It also means that the algorithm can work well on small to medium-sized datasets. Because ϵ has to be set manually, SVMs are sensitive to feature scales.[cf. 6, p. 154]

If the data can not be represented by a linear model, a kernelized SVM model has to be used.[cf. 6, p. 154] The idea behind this approach is similar to polynomial regression: By adding new features that are combinations of the input features and powers of those, the model can tackle non-linear tasks and discover new relationships among features.[cf. 6, p. 154] To deal with overfitting caused by a high polynomial degree, regularization can be applied.[cf. 6, p. 154]

2.3.5 Decision Trees

"Decision trees work by successively splitting the data into smaller segments until all of the target variables are the same or until the dataset can no longer be split."[7, p. 180] By splitting the data over and over again, a tree-like structure is formed. This tree can then be used for classification or regression.[cf. 6, p. 175] If used for classification,

the trees leaf nodes contain the class the new instance would be predicted to be.[cf. 6, p. 168] If used for regression, the leaf nodes instead contain the mean target value of the instance of the training set that is associated with that leaf node.[cf. 6, p. 175]

In the building process, the algorithm automatically selects the most important features and puts them on top of the tree.[cf. 10, p. 74] A decision to split represents the best choice at a given time and does not take the global optimal into account.[cf. 7, p. 180] Therefore the algorithm is considered greedy.[cf. 7, p. 180] Because the top levels have a higher impact on the prediction made by the tree, generalization goes down if the new instance "[...] doesn't follow exactly the same distribution as the training set [...]".[10, p. 74]

2.3.6 Random Forest

Random forest is part of a technique called *ensemble learning*, which uses the concept of "wisdom of the crowd".[cf. 6, p. 181] The concept implies, that aggregating a large number of answers or predictions of different individuals or models will lead to a better prediction.[cf. 6, p. 181] To overcome the lack of generalization mentioned in 2.3.5, random forest builds an ensemble of trees, each trained on a different subset of the data to result in a variety of tree structures.[cf. 10, p. 74, cf. 6, p. 181] A classification can then be made by picking the class the majority of the trees predicted.[cf. 10, p. 74] If used for a regression task, the mean of the trees predicted values is taken instead.[cf. 10, p. 74] By introducing building trees on random subsets of the training data, the best feature among the subset is chosen instead of the global best feature. This leads to better generalization by trading a higher bias for lower variance.[cf. 6, p. 189]

To further improve ensemble learning methods, *boosting* can be used. The idea behind it is to sequentially train the predictors with each predictor trying to improve its predecessor.[cf. 6, p. 191] Literature talks about two popular methods of boosting. *AdaBoost* or *adaptive boost* tries to improve the predictors by adjusting the dataset.[cf. 14, p. 209] This is done by first training a base classifier like a decision tree on a random subset of the training data.[cf. 14, p. 209] After training, predictions on the training set are made and each sample is given a weight according to the prediction's outcome.[cf. 6, p. 192] Misclassifications increase the weight of a data instance and correct classifications decrease it.[cf. 6, p. 192, cf. 14, p. 209] Then the next classifier is trained on a random subset of the training data prioritizing the incorrectly classified instances.[cf. 14, p. 209] This process is repeated until a defined number of classifiers is reached. Again, predictions can be made by combining the predictions of all classifiers. If the ensemble is overfitting the training set, the number of estimators can be reduced or strong regularization can be applied to the base estimator.[cf. 6, p. 195] Depending on the underlying models, AdaBoost can also perform regression tasks.[cf. 14, p. 209] A downside of this method is that due to the sequential training

of the models, parallelization is not possible, slowing down the training and reducing scalability.[cf. 6, p. 193] Another boosting method is *gradient boosting*, which "[...] tries to fit the new predictor to the residual errors made by the previous predictor."[6, p. 195] The residual errors are the differences between the expected values and the predictions made by the estimator.[cf. 14, p. 212] [Hearty 14] mentions a library called *XGBoost* that implements gradient boosting and has won several data science competitions. The documentation of XGBoost states that it is optimized and distributed, leading to high efficiency, flexibility and portability.[cf. 15]

2.3.7 Artificial Neural Networks

Artificial Neurons

An Artificial Neural Network (ANN) is a concept of machine learning that is based on the biological neurons in the human brain.[cf. 16, p. 8] It consists of artificial neurons that share connections between each other to form a network. ANNs can be used for a variety of tasks including regression, classification or transformation of the data.[cf. 14, p. 28]

An artificial neuron takes the sum of the inputs $x_1, x_2, ..., x_n$ that are multiplied by their corresponding weights $w_1, w_2, ..., w_n$ and then applies an activation function to it.[cf. 16, p. 8] In many cases a bias term b is added to the sum.[cf. 16, p. 8] The activation function then uses the sum to determine whether or not the neuron will fire and what value will be fired. The output y can be calculated by applying the activation function $f()$ to the sum of the bias term b and the product of the input vector $\boldsymbol{x}$ and weight vector $\boldsymbol{w}$.[cf. 16, p. 8]

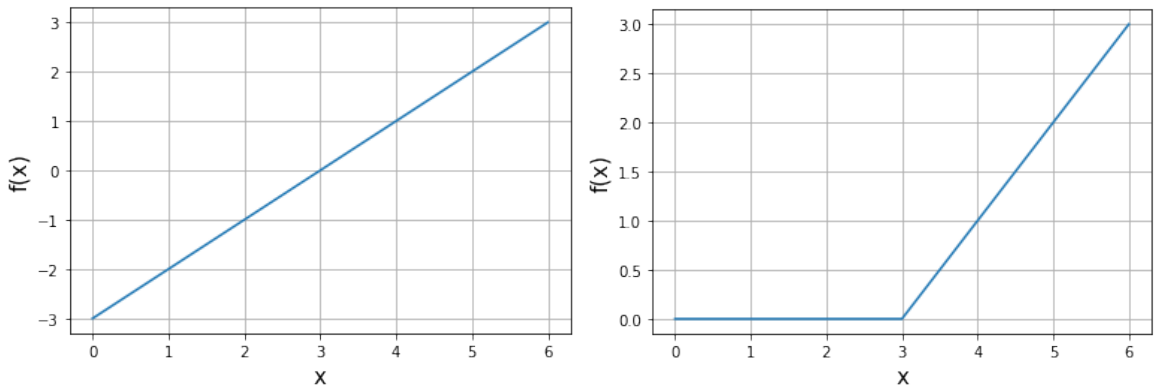
$$y = f(x * w + b) \tag{2.13}$$

Activation Functions

Depending on the task the network should perform, different activation functions can be used. Most of the time the hidden layer(s) all have the same activation function with the output layer differing.

The *linear* function is shown in 2.7a and acts like an identity function.[cf. 12, p. 66] This means that the input is passed through the neuron without being changed.[cf. 12, p. 66] Since there is no limitation to the value, this function can be used for regression because continuous values have to be predicted.[cf. 6, p. 272] [Géron 6] mentions that for regression task, no activation function should be used in the output layer, which

refers to the linear function passing through the value without modifications.[cf. 6, p. 272]



(a) Linear Function (b) Rectified Linear Function

Figure 2.7: Activation Functions

The *rectified linear* or *ReLU* function is similar to the linear function. "While the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship [...]".[cf. 12, p. 69f.] [Géron 6], [Patterson 12] and [Nikhil & Locascio 16] state that the ReLU function is the preferred activation function for the hidden layers and is also faster to compute than some other functions. The function can be seen in 2.7b.

Feed Forward Neural Network

A very common combination of such artificial neurons in supervised learning is a three-layer feedforward neural network.[cf. 14, p. 30] Neurons are split into layers and are only connected to the neurons of the next layer.[cf. 14, p. 30] The first or input layer takes the input and forwards them to the second layer, the hidden layer.[cf. 14, p. 30] It is during the forwarding process the weights are applied to the input data.[cf. 14, p. 30] Then the neurons in the hidden layer use their activation functions to determine the output values. These are then communicated to the third layer, the output layer.[cf. 14, p. 30] The task of the output layer is to deliver the network's results, which is done by applying an activation function that returns the output values in the expected value range.[cf. 14, p. 30] Additional hidden layers can be added to enhance the ANNs ability to discover complex connections and patterns between the data.[cf. 17, p. 158f.]

The network "learns" by adapting the weights of each input received by a neuron. To deal with the high dimensional space of the weights caused by the number of connections between the layers, the *backpropagation* algorithm is used.[cf. 16, p. 23] First, the error is measured by comparing the prediction to the desired output, then the error contribution from each connection is evaluated by going through each layer in reverse.[cf. 6, p. 261f.] Finally the gradient descent algorithm mentioned in 2.3.2 is used to calculate the gradient and adapt the weights accordingly.[cf. 6, p. 261f.]

An ANN with zero hidden layers is "[only] capable of representing linear separable functions or decisions."[17, p. 158] Adding hidden layers extends this capability by the approximation of "[...] any function that contains a continuous mapping from one finite space to another."[17, p. 158]

Deep Neural Networks

"When an ANN has two or more hidden layers, it is called a Deep Neural Network (DNN)."[6, p. 261] Depending on the number of layers, the network can represent data of different complexities. [cf. 17, p. 158]

A DNN with two hidden layers "[can] represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy."[17, p. 158] Adding more hidden layers to the network will increase the complexity that can be represented but also increases the risk of overfitting the training data. [Heaton 17] also mentions that two hidden layers are more than enough to "[...] represent functions with any kind of shape [...]" and there is "[...] currently no theoretical reason to use neural networks with any more than two hidden layers."[17, p. 158]

3 Concept Development

3.1 Actual State Analysis

The most important product of the Knauf group is the gypsum plasterboard. They are produced in plants all over the world. The first step in the production cycle is crushing the crude gypsum $CaSO_4 2H_2O$ or dihydrate into small pieces. This is done in the crusher followed up by a subsequent milling process in the mill. The product is then calcined in the kettle where it becomes stucco, $CaSO_4 \frac{1}{2} H_2O$ or hemihydrate. It is then mixed with water and various additives to improve the attributes of the final product. To bring the gypsum into form, it is spread on the face paper and covered by the back paper while running on the boardline. Several sensors make sure that a product with uniform thickness and high quality is produced. The endless board is then cut into different sizes at the shear. To harden the gypsum, the boards are sent into a dryer to get rid of the additional moisture. After the drying process, the boards are trimmed and stacked on pallets, packaged and brought to the warehouse for storage. The production process with its parts is visualized in 3.1.

In this work, the focus lies on the dryer and related sensors. The dryer at the location of choice has four zones with different airflow and heat. The boardline moves the plasterboards through all four zones until they have lost their moisture. Because a lot of heat is lost to the surrounding, attempts like reusing exhaust temperature are made to improve efficiency. Being the most energy-consuming part of the plant, with approximately 45MWth in comparison to 20MWth required for the initial drying process in the mill and the calcination process combined, it is of high interest to the Knauf group to improve the dryer's energy efficiency.[2, p. 3] Due to the high complexity caused by the size and surroundings of the machine, manual modeling is nearly impossible. To handle this complexity, the plant is equipped with several sensors constantly measuring temperature, pressure and many more influencing figures. To join this data, ResMa is used.[18] For a better overview, sensors can be organized in a hierarchical structure representing different steps in the production process or physical boundaries. Additionally, key figures using the data gathered by the sensors can be created to perform further calculations automatically. Users can access this data by logging into a web interface which is only accessible inside the companies local network. Each plant has its own hosting server which is based at the respective plant location to ensure a fast and non-stop recording. To store the data, ResMa uses an SQL database. To further increase the value of the web interface, users can plot data in charts to manually control data quality and perform simple visual analysis.

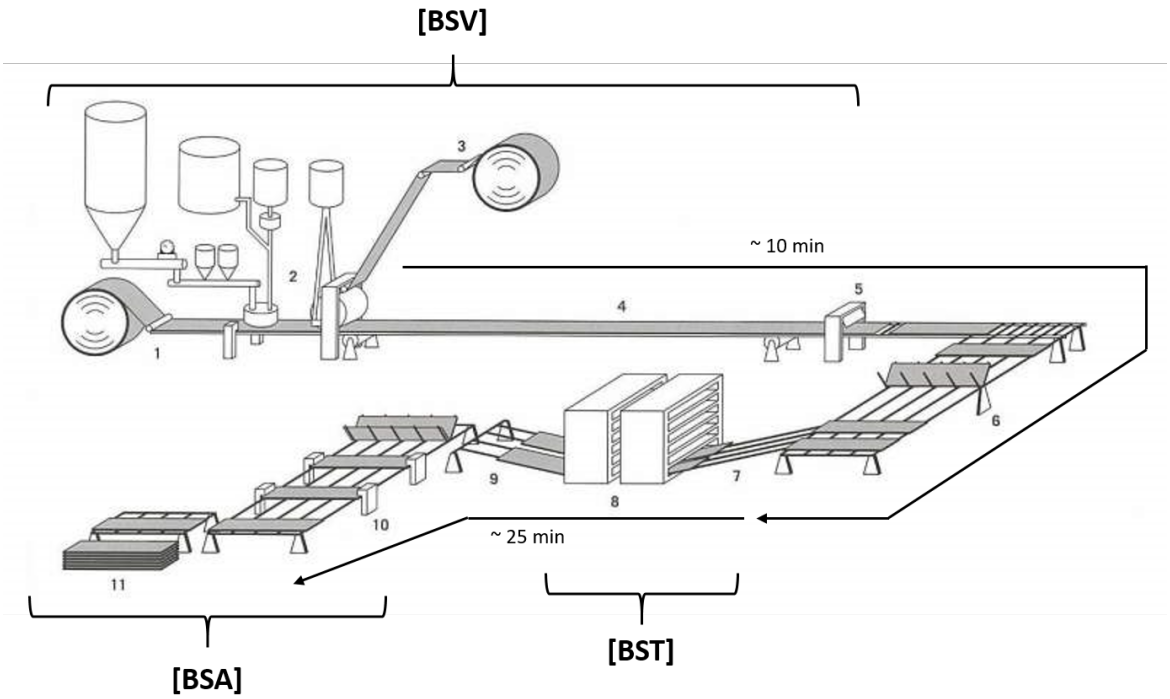


Figure 3.1: Gypsum Boardline - Source: [2, p. 3].

1	Stucco and Additives	7	Dryer Inlet with Tipple
2	Mixer	8	Dryer
3	Back Paper	9	Dry Cross Conveyor
4	Face Paper	10	Finished Boards
5	Shear	11	Removals
6	Booker / Flipper		

3.2 Methodology

The actual state analysis shows that a lot of data involving the dryer is available. Because of the high complexity of the machine and its surroundings, a manual analysis is not a viable option. In the past decades, machine learning has become more and more popular due to the fact that an enormous amount of data became available and the processing power reached a point where it can be analyzed. The techniques are also capable of finding connections and relationships between data when the amount of data and complexity can no longer be grasped by the human brain. Looking at the available data regarding the dryer, machine learning seems like a good approach to tackle the problem. Furthermore, the CRISP-DM provides a guideline for performing such an analysis without losing the focus on the business objectives. On the basis of the preceding analysis, the following hypothesis can be formulated:

> "**By performing the CRISP-DM and utilizing machine learning, valuable new insights into the factors influencing a plasterboard dryers energy consumption can be gained from the data recorded by the ResMa.**"

Based on CRISP-DM, first a concept is developed in which the further course of action is justified. Furthermore, methods are introduced that support an execution of the steps proposed by [Wirth & Hipp 4]. The concept is then implemented and evaluated. The resulting models are compared and ranked with regards to the chosen performance metrics. Insights are assessed in terms of business value and the results are used to accept or reject the hypothesis.

3.3 Business Understanding

3.3.1 Business Objectives

In the actual state analysis, the dryer was identified as the most energy-consuming part of the production plant. It also shows a high potential of reducing the energy consumed and therefore to reduce cost and environmental impact. In conclusion, the business objective is to decrease the energy consumption of the dryer and identify reasons for fluctuation in energy efficiency. Because this is a quite complex problem and the time frame of the project is limited, it is further split into subproblems. One of those is to understand the relationship between influential factors and energy consumption. Another one is to analyze how the type of plasterboard correlates with the energy needed for the drying process. As mentioned before, the plant is already equipped with the ResMa, providing sensor data connected to this problem. Because of the

large amount and high complexity of the data, manual analysis can only partially be performed by the business, mainly by utilizing the web interface to plot the data. By using a machine learning model, the business is looking to gather insights that manual processing can not extract. Using this model, the business should be able to test their assumptions regarding energy consumption.

It should also be investigated whether machine learning is the appropriate tool to use for this type of analysis. Furthermore, the process is seen as a case study for introducing data mining practices to the organization. This includes analyzing the suitability of the data collected by the sensors and possible improvements that can be made to better design the process in the future. The objective is to set the foundation for future data mining projects and analysis on a larger scale.

To complete the analysis of business objectives, the type of data mining problem has to be defined. Because the energy consumption is most likely represented by a floating-point number and data already containing those values is given, the problem can be identified as a supervised learning task, to be more precise, a regression. These findings restrict the process of concept development by limiting the number of techniques and algorithms one can choose from. It is due to this impact that parts of the business understanding phase have to be performed at the beginning of the concept development phase.

3.3.2 Platform Selection

When turning to machine learning, the goal of the business partners usually is to get insights about relationships in data, to analyze and to make predictions. To perform such tasks, data has to be collected and made available by the business. The disclosure of data outside their sphere of influence can raise concerns. Also, ambiguities about further use are prone to misunderstandings. Knowing the goals and what concerns exist or might arise is key to this phase, because it sets the foundation for the data extraction and the computing environment.

Machine learning algorithms can consume a lot of computational resources. As described in 2.3, models can have different complexities, can be trained multiple times to improve the performance using grid search or cross validation and scale differently depending on the size of the data. This is why a lot of businesses move to the cloud where e.g. *Amazon Web Services*, *Microsoft Azure* or *Google Cloud Platform* offer the resources needed to perform those calculations in less time.

Due to concerns about security and a lack of understanding, the cloud is not widely accepted amongst most business partners. This is closely connected to the confidentiality of the data and the data governance of the IT department. To come to an agreement that serves both sides, a consensus about the location the data is stored

and processed at has to be achieved. This state can be reached by discussing concerns with the business and evaluating the confidentiality of the data. Also, benchmarks and a comparison of different solutions should be provided to support the process.

The process of implementing those requirements is developed in the course of this thesis and consists of four steps. In the first one, an estimation of the required resources is made utilizing the benchmarks created. The second step tries to find different approaches that can satisfy these needs. In the third step, the options are evaluated and compared to the security guidelines of the company. Compliant options are then presented to the business in the last step. It is this step where a consensus is created and an agreement should be found. A visualization of the process is shown in 3.2. The report about the required resources that was presented to the business can be found in the appendix under A.1.2.

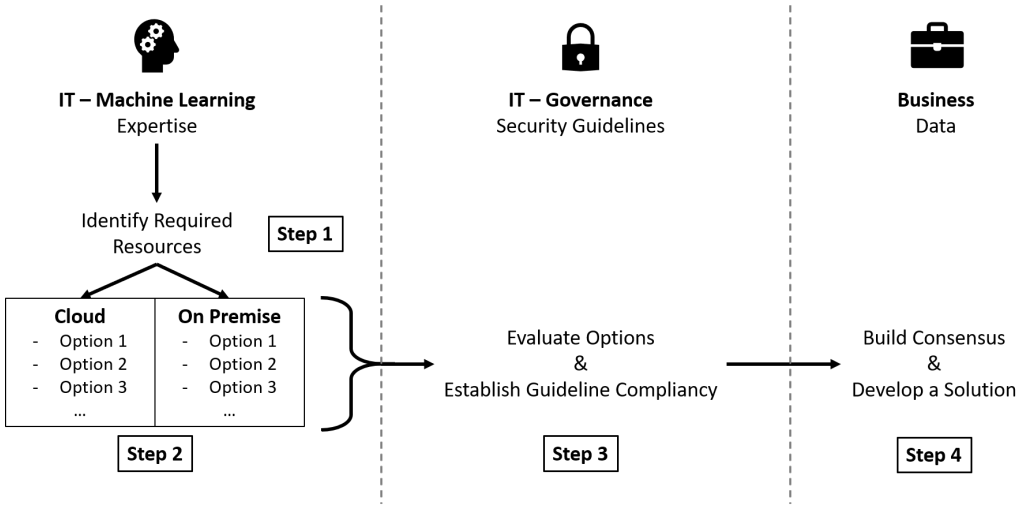


Figure 3.2
The Process of Selecting a Platform together with the Business

3.4 Data Understanding

3.4.1 Data Export

Before extracting the data from the ResMa, a pre-selection of features has to be made. This is done in cooperation with the plant engineers, providing a list of attributes that are expected to have an impact on energy consumption. This way, the number of features can be reduced to its most essential selection. Then a chart can be created using the web interface of the system. A chart can consist of different attributes and the selection of compression strength. The compression can reach from seconds and

minutes over the quarter of an hour up to hours, days, months and years. After these selections have been made, the data is plotted in a graph, enabling the user to spot holes in the record before exporting. If no missing data is spotted, the underlying data can be exported as a 'CSV' file which stores the data as comma-separated values.

Before being able to further analyze the data, an environment has to be set up to perform actions on the data. Depending on the developed solution, contracts with a cloud provider have to be set up or the internal IT department has to provide hard- and software fitting the requirements.

3.4.2 Exploratory Data Analysis

After setting up the environment, an Exploratory Data Analysis (EDA) can be performed. According to [Yip, Groenen & Nalbantov 19], the term EDA was coined by Tukey in 1993. In his work, a precise definition is hard to find and the term is only described vaguely.[cf. 20, Data Analysis, Exploratory] Though there is no such definition, literature offers a variety of attempts to do so:

> "EDA can be defined as the art and science of performing an initial investigation on the data by means of statistical and visualization techniques that can bring out the important aspects of the data that can be used for further analysis."[cf. 19, p. 1f.]

> "Exploratory data analysis is the process of exploring your data, and it typically includes examining the structure and components of your dataset, the distributions of individual variables, and the relationships between two or more variables."[cf. 21, p. 31]

Therefore EDA uses statistical and visualization techniques to explore the data and its relationships. These structural findings can then be used to perform a deeper analysis of the data. The goal of such an analysis is to determine whether there are problems with the dataset like outliers or anomalies and if the questions and goals defined at the beginning of the data mining process can be answered and reached by the data obtained.[cf. 21, p. 31, cf. 22, p. 9f.] Furthermore, valuable assumptions for the modeling process can be identified.[cf. 21, p. 31, cf. 22, p. 9f.]

> "If one is not using statistical graphics, then one is forfeiting insight into one or more aspects of the underlying structure of the data."[22, p. 11]

This quote by [Jaggi 22] underlines the importance of a visualization of the data and that one might experience big drawbacks if not using such methods. He further describes in [22, p. 11] that "[those] are the shortest path to gaining insight into a dataset in terms of [...]"

- Testing assumptions
- Model selection and validation
- Estimator selection
- Relationship determination
- Outlier detection

In more particular, graphical techniques in EDA include plotting of the raw data in the form of e.g. histograms, probability plots and scatter plots, exploring simple statistics like e.g. mean plots and standard deviation plots but also to position those plots wisely to support the pattern-recognition abilities of us humans.[cf. 22, p. 10]

In the CRISP-DM cycle, the EDA is a very important technique that can be used in the data understanding phase to get deeper insights.[cf. 19, p. 2] In this thesis, EDA is used to support the data description as well as the exploration report. The findings can then be used in the data preparation process and later help to select the models that can benefit most from these findings and assumptions.

3.5 Data Preparation

3.5.1 Data Cleaning

> "Garbage in, garbage out."[cf. 6, p. 25, cf. 14, p. 155]

This quote is often mentioned in literature when talking about data preparation. It means that if the input data is of poor quality, the model output will also be of poor quality. Care must be taken to ensure data quality as high as possible. Improvements to the data quality can be made by cleaning up the data, selecting relevant features and engineering of additional features.

Since one cannot simply assume that the data received is of high quality and all values are plausible, data cleaning must be carried out on the dataset. Also most machine learning algorithms break if values are missing.[cf. 6, p. 60] [Géron 6] proposes three different options of how to deal with missing values:

- "Get rid of the corresponding sample / row"
- "Get rid of the whole attribute / column"
- "Set the values to some value (zero, the mean, the median, etc.)"[6, p. 60]

The process of data cleaning also uses insights gathered in the data understanding phase to identify inaccurate or dummy values and identify outliers.[cf. 12, p. 326] Such values can occur due to sensor malfunction and human or computational errors.

3.5.2 Feature Engineering and Selection

The process of feature engineering aims to derive new features from domain-specific and data type-specific knowledge.[cf. 14, p. 129] In more detail, mathematical transformations are used to transform the raw input data.[cf. 10, p. 107] E.g. features can be combined by calculating a key figure representing the relationships or dependencies between those.[cf. 10, p. 107] It also offers the opportunity of bringing in outside knowledge into the model by calculating e.g. key figures beforehand.[cf. 10, p. 108] Feature engineering is also very helpful when unstructured data like e.g. time series, images, video or text is available.[cf. 10, p. 108] Relevant data can then be extracted and transformed into a vectorized form that can be used as a feature.[cf. 10, p. 108] One technique to accomplish this task is to encode those features by mapping them to integer numbers that the algorithm can handle.[cf. 6, p. 62] If the categories contain some sort of similarity with each other, one-hot encoding can be used.[cf. 6, p. 63] This approach creates a new feature for each category and sets the value to 1 if the sample is part of this category or 0 if it is not.[cf. 6, p. 63] It has to be mentioned that a decision has to be made in respect to the feature and encoding is just one way to transform unstructured data to a vectorized form. The engineering of features can also be used to improve the interpretability of the trained model.[cf. 10, p. 108] Sometimes features are very abstract making it hard to interpret the model's predictions.[cf. 10, p. 108] To solve this issue, new features can be created that are interpretable more easily.[cf. 10, p. 108] Some models are able to handle a lot of features at once and can also identify the most important features, mostly ignoring the unimportant ones.[cf. 10, p. 108f.] When using such models, like e.g. lasso or ridge, feature engineering should be used to create a large set of features and to leave it up to the model which ones to use.[cf. 10, p. 108f.] To sum it up, by using feature engineering, relationships that can easily be identified by utilizing domain knowledge or other related skills can be fed to the algorithm.[cf. 10, p. 107] Ultimately, this can lead to better model performance.

Feature selection, on the other hand, is described by [Géron 6] as "selecting the most useful features to train on among existing features."[cf. 6, p. 26] To select those "most important features", different techniques can be employed. One method is to select features based on mutual information, relying on the assumption that "[in] reality, good features should not only be strongly correlated with class labels but also should not be highly correlated with each other."[23, p. 94:12] This means that if two features show a high correlation to each other and to the target, both may contain redundant information that allows only one to be selected. If the model used is not capable

of such an automatic selection of the features, feature selection has to be performed manually.

3.5.3 Feature Scaling

Because most machine learning algorithms have problems dealing with data at very different scales, feature scaling has to be performed on the data.[cf. 6, p. 65] Two common methods are mentioned in the literature, one being standardization and the other one being normalization.[cf. 6, p. 65, cf. 7, p. 29, cf. 14, p. 157]

Normalization or *min-max scaling* rescale the values so they all end up in a range from 0 to 1.[cf. 6, p. 65] The largest value is represented by 1 and the lowest value is represented by 0 with all other values lying in between in respect to the proportions to the largest and smallest value in the original range.[cf. 14, p. 158]

When using *standardization* or *standard scaling*, first the mean value is subtracted from the values, resulting in a zero mean of the standardized data.[cf. 6, p. 65] Then the values are divided by the variance to achieve unit variance.[cf. 6, p. 65] It is important to mention that the values are not capped to a specific range like normalization does.[cf. 6, p. 65] This can lead to problems when training neural networks, as some of them expect values in the range of 0 to 1.[cf. 6, p. 65] An advantage of standardization is though that it is less sensitive to outliers.[cf. 6, p. 65]

3.6 Modeling

3.6.1 Algorithm Selection

The NFLT in 2.3.1 states that the ideal algorithm can not be determined beforehand. As proposed earlier, assumptions can be made to narrow down the models that most likely lead to a working solution. The first assumption that can be made is that the model should be capable of performing regression tasks. Because the problem centers around energy consumption, classification does not seem like a good approach. Energy consumption or efficiency is typically represented by physical key figures which usually are floating-point numbers. To predict such a regression is required. Because the business provides the information that multiple attributes represent the energy consumption, the task can further be classified as a multi-output regression, predicting multiple output values at once.

The second assumption that should be considered is that the relationship of the data is unlikely to be linear due to the high complexity expected. A reasonable decision

should be to discard linear models from the selection. However, the benchmark in A.1 shows that training a linear model seems to consume a lot less time in comparison to a more complex model. Because of this, all linear models discussed in the theoretical foundation are included in the final selection. As the number of samples is always expected to be higher than the number of features, elastic net should only be considered over lasso regression if the EDA reveals high correlations between the features. Otherwise, ridge and lasso regression are added to the selection, because dimensionality reduction is wanted and has a supportive effect on presenting gathered insights to the business by reducing complexity. They are further compared to linear regression to determine whether regularization leads to an improvement in model performance. If there is indeed no linear relationship, polynomial regression and kernelized SVM is added to the algorithm pool to cover the case. Praised in literature, ensemble learning techniques like random forest and the XGBoost algorithm are also selected. To conclude the process of algorithm selection, ANNs and DNNs are added due to their ability of modeling even higher complexity. A detailed description of the libraries and classes used in particular is attached in A.4.1 and A.4.2.

3.6.2 Model Training

Training Process

Training a model can differ depending on the underlying algorithm used. The libraries selected, provide mostly unified interfaces that make the training process easier to understand. Each model has a *fit*-method that takes the input samples as matrix X and the expected output vector $\boldsymbol{y}$. By calling the method, the data is processed by the algorithm which tries to minimize a loss function by adapting the model's weight parameters $\boldsymbol{w}$. It is said that the model tries to *fit* the data to represent it as good as possible. After the training process is finished, a scoring method can be called to measure the model's performance. The implementation of this method depends on the performance metric and the type of task performed. It takes a separate dataset with the same number of features and evaluates the performance on new samples that the model has not seen before. Furthermore, a *predict*-method is offered that can be used to produce the output for a certain input by applying the weight parameters to the data.

The standard approach of training is to first call the *fit*-method. Then the *score*-method is used to see how well the model performs on new data. If an acceptable model performance is reached, the *predict*-method can be used to make predictions and take the model into production.

Bias-Variance-Tradeoff

The error that a model makes on new data is called the generalization error or out-of-sample-error.[cf. 6, p. 29] It is a metric for how well a model performs on instances it has not seen before.[cf. 6, p. 29] Literature defines a model's generalization error as the sum of three components: *bias*, *variance* and the *irreducible error* or *random noise*.[cf. 7, p. 172, cf. 6, p. 126f.] Bias occurs due to wrong assumptions in data complexity.[cf. 6, p. 126f.] For example if it is assumed that the data is linear, but in fact, it is quadratic, the model will have a high bias value.[cf. 6, p. 126f.] A result of a high bias is underfitting the training data.[cf. 6, p. 126f.] This means that the model's complexity is not able to learn the underlying structure of the training data.[cf. 6, p. 28] [Geron 6, p. 126f.] states that underfitting can be fixed by:

- Selecting a more powerful model that provides more complexity in the form of more parameters
- Using feature engineering to feed better features to the learning algorithm
- Reducing the constraints of the model by reducing the regularization hyperparameter, if the model is regularized

If a model is too complex in comparison to the training data's complexity or noisiness, high variance is a result.[cf. 6, p. 27] The model then has "[...] excessive sensitivity to small variations in the training data."[6, p. 126f.] This means that it performs well on the instances in the training set, but struggles generalizing on unseen data. [Geron2017 Page 26] This phenomenon is called overfitting.[cf. 6, p. 28] [Geron 6, p. 27] offers possible solutions to reduce overfitting:

- Simplify the model by selecting one with less complexity in the form of fewer parameters, by reducing the number of features in the training data or by constraining the model by e.g. applying regularization
- Gather more training data
- Reduce the noise in the training data by e.g. removing outliers and fixing data errors

The irreducible error can not be reduced by modifying the model. This part of the error is a result of the noisiness of the data itself.[cf. 6, p. 126f.] It can only be reduced by a data cleanup.[cf. 6, p. 126f.] This process involves e.g. fixing the data sources, such as broken sensors, reducing data errors and removing outliers.[cf. 6, p. 126f.] "Increasing a model's complexity will typically increase its variance and reduce its bias. Conversely, reducing a model's complexity increases its bias and reduces its variance."[6, p. 127] In machine learning this is known as the *bias-variance tradeoff* and is present in many areas of the field.[cf. 6, p. 127, cf. 7, p. 172]

Training, Test and Validation Set

To now see how well the model generalizes and to estimate the generalization error, it has to be tested on new instances.[cf. 6, p. 29] To do so, the original dataset is split into a training set and a test set before starting the training process.[cf. 6, p. 29] As the names imply, the training set is used to train the model and the test set is used to test the model using the generalization error as a metric.[cf. 6, p. 29] By analyzing the errors on both sets, overfitting can be identified. This is the case when the training error is low, the model makes few mistakes on the training set, and the generalization error is high, the model makes many mistakes on the test set.[cf. 6, p. 29] A common split is using 80% of the original dataset as the training set and to hold back 20% for the testing process.[cf. 6, p. 29] Neural Networks usually are way more complex than the other algorithms mentioned, which means they have a higher chance of overfitting.[cf. 9, p. 19f.] To make sure this doesn't happen, the training error has to be monitored to stop training before the overfitting starts.[cf. 9, p. 19f.] A similar problem occurs when training the other algorithms mentioned.[cf. 6, p. 30] When tuning hyperparameters, it is possible that they work specifically good on the chosen training set.[cf. 6, p. 30] To prevent this, the generalization error should also be monitored during training.[cf. 6, p. 30] Because the test set should be used to estimate the generalization error of the final model, a new way has to be found to monitor the training error during the training process.[cf. 9, p. 19f.] Therefore an additional split is performed to produce a validation set.[cf. 9, p. 20f.] To guarantee a reasonable size of all three sets, [Marsland 9, p. 20f.] suggests using a 50:25:25 (50% training, 25% validation, 25% test) split if enough data is present. If not, a 60:20:20 (60% training, 20% validation, 20% test) split should be used, sacrificing test and validation data in exchange for a larger training set. He also points out the importance of randomly reordering the original dataset before splitting it.[cf. 9, p. 20f.] This way it is guaranteed, that the order of the original dataset is not influencing the representation of the data in each set.[cf. 9, p. 20f.]

It can be seen that using a validation set for the evaluation is very data-intensive. To avoid "wasting" too much training data on the validation process, a technique called *cross validation* can be used.[cf. 6, p. 20] [Marsland 9] describes the same concept in more detail calling it "leave-some-out multi-fold cross-validation". [Marsland 9, p. 20f.] The training set is randomly split into partitions or subsets. The amount of those can be specified by parameter K. This is why the technique is also called "K-fold cross-validation".[6, p. 69.] The selected algorithm is then trained on those subsets, leaving out one subset that is used the as validation set.[cf. 9, p. 20f.] In the next step, the same model is trained on a different combination of the subsets.[cf. 9, p. 20f.] After training a model for every combination, the model type and hyperparameters that performed best on the validation sets can be identified.[cf. 6, p. 30] This selection is then trained on the training set and the generalization error is measured using the test

set.[cf. 6, p. 30] By increasing the number of models trained to the number of possible subset combinations, a tradeoff between data and computation time is made.[cf. 9, p. 20f.]

3.6.3 Model Evaluation

To evaluate a trained model, its ability to generalize on new data has to be measured. This can be done by different performance metrics, taking into account the expected value in comparison to the predicted value. Because this project focuses on a regression task. Only a limited variety of different metrics can be used. Due to the limited scope of this project and preferences that are mentioned by literature as commonly used, three performance metrics are discussed and selected for further use. It has to be noted that for unification purposes, the equations in this section are a combination of the notations provided by [12, Patterson, & 6, Géron] and supported by the scikit-learn documentation [24].

To measure the performance of a regression model, [Géron 6] suggests using the Root Mean Squared Error (RMSE). This metric represents the standard deviation of the errors the model makes on predictions.[cf. 6, p. 37.] The distance of the data points to the line fit through the data is squared, summed up and divided by the number of instances. Then the square root of the result is taken. The equation can be seen in 3.1. He also notes that in practice, the MSE is easier to minimize improving model performance.[cf. 6, p. 107] Because minimizing the MSE also reduces its root, the RMSE can still be used to evaluate the final model.[cf. 6, p. 107] This coincides with [Patterson 12] stating "[when] working on a regression model that requires a real-valued output, [...] the squared loss function [is used] [...]".[12, p. 72]

$$RMSE(\hat{y}, y) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2} \tag{3.1}$$

If outliers are expected, the Mean Absolute Error (MAE) should be calculated instead.[cf. 6, p. 39] The preceding errors all use the square of the difference between the actual value and the target value. Contrary to this, the MAE uses absolute error instead. This means that for outliers that produce a higher difference, the MSE and RMSE produce significantly higher errors by squaring the difference. If many outliers exist, this can falsify the model's progress.

$$MAE(\hat{y}, y) = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i - y_i| \tag{3.2}$$

Another important metric that is often used in the scikit-learn package to train and evaluate models is the R Squared (R^2) or the *coefficient of determination*.[25] The numerator consists of the sum of the squared differences between the labels $\boldsymbol{y}$ and the model's predictions $\hat{\boldsymbol{y}}$. The denominator is the sum of the squared differences of the actual data $\boldsymbol{y}$ from the mean $\bar{y}$. The equation can be seen in 3.3. When the model generalizes well on new instances, the numerator will be smaller than the denominator, resulting in the fraction being small, assuming the estimator is performing better than the mean. On the other hand, if the model is performing poorly, the numerator will be big, leading to a big fraction value. Because the fraction value is subtracted from 1, the smaller the fraction value, the higher the score. It has to be mentioned that if the predicted values are worse than the mean, the fraction value can be above 1, leading to a negative score. To sum it up, the score should be as close to 1 as possible, meaning the model generalizes very well on new instances. If the score is close to or below 0, the model generalizes poorly.

$$R^2(\hat{y}, y) = 1 - \frac{\sum_{i=1}^{m} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{m} (\bar{y}_i - y_i)^2} \tag{3.3}$$

3.6.4 Hyperparameter Tuning

After the models are trained and a first evaluation was performed, the models promising the best performance should be fine-tuned for further improvements. A way to do so is applying a concept called *grid search*. Models have different hyperparameters that are set before the training begins and that can affect the training phase itself as well as its result. Grid search takes different values for each hyperparameter and trains a model for every combination. To effectively perform a grid search, an evaluation metric has to be chosen that the search tries to maximize or in case of an error to minimize.[cf. 10, p. 101] For regression the R^2 is chosen because the closer to 1, the better the generalization capability of the model.[cf. 10, p. 101] Next, an algorithm and the hyperparameters to fine-tune are selected.[cf. 10, p. 101] The hyperparameters are usually organized in a dictionary containing the different values for each parameter that should be tested.[cf. 10, p. 101] This dictionary is sometimes called *parameter grid*. It has to be noted that parameters can also be part of the feature selection step, meaning that the number of selected features can be fine-tuned. If previous experience with the value selection exists, [Géron 6] suggest "[...] [trying] out consecutive powers of 10 [...] or a smaller number if [...] a more fine-grained search [is wanted]".[6, p. 72] Another promising approach is using the default values provided by the libraries as a starting point and evolving the grid around those. After the selection is made,

each combination is used for the training step. Usually, grid search is combined with cross validation to find the parameters that generalize best on new instances and to guard against overfitting.[cf. 10, p. 101] After training, the means of the model's scores are taken, compared and the model with the best score is chosen.[cf. 10, p. 101] This model than has the highest likelihood of generalizing well on new data.[cf. 10, p. 101]

As seen in 3.4, the number of models trained by grid search can be calculated by multiplying the number of values for each parameter and parameter K representing the number of folds used in the cross validation.[cf. 6, p. 74] When the number of values or parameters increases, the number of models that have to be trained rises sharply, leading to high computational resources being required. To reduce those and to maintain control over the computing budget, a variation of grid search, the *randomized search* can be used instead.[cf. 6, p. 74] This technique chooses a set number of random parameter combinations from the grid and trains models on those, again using cross validation.[cf. 6, p. 74] The number of combinations can be specified by a parameter. Even though there is a tradeoff between model performance and training time, randomized search becomes inevitable when working with a large parameter grid and usually converges to a good solution. A promising approach would also be to perform a randomized search on the promising models and then again select the most promising, performing exhaustive grid search only on those.

Figure 3.3: Number of Models Trained Depending on the Parameter Grid

$$n_1 * n_2 * n_3 * ... * n_p * k \tag{3.4}$$

n number values for parameter p
k number of folds used for cross validation

When applying grid search to ANNs or DNNs, the number of neurons of each hidden layer and the number of hidden layers can also be included in the grid. [Heaton 17] mentions that too many neurons can lead to overfitting and too few neurons can lead to underfitting and it is therefore key to find the space in between.[cf. 17, p. 158] He also warns that "[an] inordinately high number of neurons in the hidden layers can increase the time it takes to train the network [...] up to the point that it is impossible to adequately train the neural network."[17, p. 158f.] According to him, there are rule-of-thumb methods that help to determine a good amount of neurons in the hidden layers.[cf. 17, p. 159] The number of hidden neurons should be:

- between the size of the input layer and the size of the output layer [17, p. 159]
- $\frac{2}{3}$ the size of the input layer, plus the size of the output layer [17, p. 159]

- less than twice the size of the output layer [17, p. 159]

A different approach named *geometric pyramid rule* is provided by [Masters 26].[cf. 26, p. 176f.] This rule states that "[...] in many practical networks, the number of neurons follows a pyramid shape, with the number decreasing from the input toward the output."[26, p. 176]. For a network with one hidden layer, the number of neurons of the hidden layer can be calculated by the square root of the number of input neurons multiplied by the number of output neurons.[cf. 26, p. 176f.]

Figure 3.4: Optimal Number of Neurons in the the Hidden Layer (three-layered network)

$$sqrt(nm) \tag{3.5}$$

n number of input neurons
m number of output neurons

Both [Masters 26] and [Heaton 17] seem to agree, that the number of neurons in the hidden layers should not exceed the number of input neurons, but should be greater than the number of output neurons. These constraints should be taken into account when selecting a parameter grid for fine-tuning the number of neurons and the number of hidden layers.

3.7 Evaluation and Deployment

After making sure that the domain knowledge provided by the business in the business understanding phase was incorporated into the process and the models were trained successfully, a final meeting with the business should be conducted. In this meeting, the results are presented by the team responsible for the project and discussed in plenary. Furthermore, it should be investigated whether or not the findings are of sufficient value for the business. By also giving an overview of the project, risks, pitfalls, and unexpected results can be identified. Those should be documented to support future data mining projects. It is in the evaluation phase that a decision about the further steps is made. If the trained models perform sufficiently, it can be moved forward to the deployment phase. Otherwise, one has to go back to the business understanding phase, iterating over an additional cycle of the process until the deployment phase can be entered.

In the deployment phase, the results are given to the business in the form of a report. In this case, an application or implementation of a repeatable data mining process is not considered as the project only covers a small part of the underlying problem and

a reusable process was not in the focus of the project. It has to be mentioned though that the report can be used in the future to develop such deployment types at a larger scale.

4 Proof of Concept

4.1 Business Understanding

Only the platform selection has to be conducted in the proof of concept because the business objectives were already developed in the concept development phase. As described in 3.2, the first step consists of identifying the resources required. To do so, a benchmark is created using dummy data to train various models and the time needed for training as reference. This way, the differences in time complexity amongst the different algorithms can be grasped more easily. It has to be mentioned though that execution time always depends on the system used and additional processes running in the background. Due to this, the result should only be used as a rough approximation of the actual runtime. The benchmark results and the process are presented in A.1. Using experience in the field, the different tasks like data exploration and preparation, modeling of simple models and modeling of DNNs can be classified by the amount of processing power, memory usage and the ability to support a *GPU* or *TPU* to speed up the training process.

After the resources are identified, different solutions can be collected and evaluated. For this project, offers from *Amazon Web Services*, *Microsoft Azure* and *Google Cloud Platform* are selected and their pricing and compute capability is listed. Furthermore, each option is matched with a task that can be performed on it. It can be seen that the only task expected to be performed on an on-premise solution is the data exploration and preparation. All the other tasks require much more computational power to finish in a reasonable amount of time. Also, the compliance policy of each company evaluated is listed for further investigation because security concerns are common among business partners. By using estimations of how long each task takes to perform, a cost estimation can be created to provide a starting point for the budgeting process. Additionally, different ways of anonymizing the data are provided to further support the need for security and to make sure that the data can not be used by third parties in case of a breach. Those practices include the removal of column names, the encoding of string values and scaling of the data. This way, the data can only be interpreted if the correct keys are available to decode e.g. column names and string values. The report created for this step can be found in A.1.2.

In the next step, the options are evaluated against the security guideline, ideally developing guideline compliance. No agreement could be found with the business because currently no standard process of evaluating the confidentiality of the data and its capability of being processed in the cloud exists. Developing such a process

requires a lot of time and does not fit into the temporal extent of this thesis. To pay respect to the scope of the work, an on-premise solution is selected for the execution of the tasks.

The proposed solution consists of an *IBM POWER8* server which is part of the company's data center and is therefore considered to be more secure. A reason for this is that every aspect can be controlled in as much detail as possible, which leads to acceptance by the security guidelines. The server is equipped with 20 *IBM POWER8* cores of the ppc64le architecture resulting in 160 threads and 256GB of memory, providing a sufficient amount of computational power. Detailed information about the operating system and technologies used is attached in A.2.1

To complete the business understanding, the performance criteria for the model has to be defined. For this project, the R^2 is chosen as it is easy to interpret and does not depend on the scale of the target values. To later evaluate the models in terms of objective fulfillment, a threshold of 0.9 is chosen. This means that if a model scores higher on the test set than this threshold, it is considered to generalize well enough to serve as a model for the business use case.

4.2 Data Understanding

4.2.1 Data Export

After finding an appropriate solution for processing, the data can be extracted from the ResMa. With the help of the responsible service owner, a prior feature selection based on domain knowledge can be performed and general findings can be gained that support further steps like feature engineering and feature selection. Because the server is stationed at the plant itself, the Internet connection becomes the *bottleneck* when downloading the data via the web interface. To achieve the selected compression strength of a quarter-hour, diagrams have to be created from which the data can be exported. These diagrams limit the amount of data shown in the mentioned compression strength to a time frame of one month. Due to these issues, data can not be exported as one single file but has to be downloaded manually as many small files. Each category represents one element of the lowest level of the hierarchy. The hierarchy is used by the system to group the inputs according to their place of occurrence, also giving the user a better overview. The attribute categories mentioned are described in 3.2. The *Zone* column indicates the section of the plant where the data is recorded. Also, the amount of files exported is given. The primary file format of the exports is 'CSV' (comma-separated values) with *artikelnummer* being the exception being exported as 'XLS' (Microsoft Excel file format) and then converted to 'CSV' using *Microsoft Excel*. Note that some of the exports include three additional files

only containing a single week, as the system automatically erases data that is in the system for a longer period of time, in this case, cutting off the first week of the first month recorded.

Category	Description	Zone	Files
abnahme	Removal	BSA	23
artikelnummer	Product produced next	-	1
trockner	General data (dryer)	BST	21
trockner_effizienz	Efficiency (dryer)	BST	22
trocknergetriebe	Gearbox (dryer)	BST	22
vorbereitungsstation	Preparation	BSV	18
waerme_rueckgewinnung	Heat recovery (dryer)	BST	23
zone01	Zone 01 (dryer)	BST	23
zone02	Zone 02 (dryer)	BST	23
zone03	Zone 03 (dryer)	BST	23
zone04	Zone 04 (dryer)	BST	23

Table 4.1: Attribute Categories

It has to be mentioned that most of the data reach from September 2017 to June 2019 with some exclusions representing less than two months. An example of this is the *artikelnummer* and the *trockner_effizienz* file. Some of the contained attributes did not exist before May 2019 and are a product of the business understanding process. During this process, it was realized that some important attributes did not yet exist in the system even though sensors for those are already placed in the plant. By connecting those to the system, the recording of the values could be started.

Because all files inside one category have the same attributes, they are merged into a single file, using the timestamps as an index, because of its uniqueness. Additionally, the numerical data uses ',' characters as decimal separators. This leads to difficulties reading in the files. Therefore the ',' characters are being replaced by '.' characters and the attributes are converted to the *float64* format afterward. The file *artikelnummer* is the only file containing an attribute of the type *string* and is therefore not converted. Also, the timestamps in each file are converted to the *datetime64[ns]* format and set as the index to make further processing easier. The file *artikelnummer* is the only one containing two timestamps. In this case, both are converted, but only the first one is assigned as the index. In the same step, the seconds of the timestamps are set to '00' to make further merging easier by avoiding mismatching due to differences in the seconds' position. This is possible because in the context of a plasterboard dryer, seconds do not have a significant impact on the data and quarter-hour exports are chosen to begin with. The rows are then aggregated on the DateTime index using a *mean* strategy in case of duplicate indices.

To handle the processing of all files, a fork of the python *multiprocessing* package called *billiard* is used to parallelize the process.[27, 28] This way processing time can be reduced significantly utilizing all of the server's processors. The difference to the original python package lies in the ability of *billiard* to also parallelize child processes of already parallelized processes. This approach is kept in mind for later use as it offers potential time savings for further steps.

4.2.2 Data Analysis

Data Description and Exploration Report

As mentioned in 4.2.1, the files are all exported in the 'CSV' format or converted to this. The files are all encoded in the 'utf-8' format and the *Time* or in case of *artikelnummer* the *Zeit* attribute is used as an index of type *DatetimeIndex*. Each file is analyzed in the following in regards to quantity, missing values, and anomalies. By analyzing the mean, standard deviation and distribution among the percentiles, assumptions about the distribution of the attributes can be made. Data that is distributed unevenly can directly impact the performance of the machine learning algorithm. The optimum would be a normal distribution among all attributes. Because this is very unlikely, attempts are made to remove outliers from the key attributes to improve their distribution. When analyzing the distributions, the so-called *empirical rule* can be used. This rule states that for a normal distribution, 99.7% of all values lie between three standard deviations away from the mean. E.g. a maximum or minimum value that does not lie in the given range or a jump between the percentiles could both be a sign for outliers being present. In the following, the absolute number and the percentage of missing or null values and additional information about the file is given. A more detailed version including the mean, standard deviation, minimum and maximum value as well as the percentiles can be found in the appendix under A.2.3.

```
DatetimeIndex: 4574864 entries, 2017-01-11 00:00:00 to NaT
Data columns (total 7 columns):     isnull    %null
Schmierung Tisch 1                 4572718    99.95
Summe Stromverbrauch Abnahme       4574864   100.00
Trockenausschuss [BSA]             4574864   100.00
Restfeuchte Platten [BSA]          4515579    98.70
Anzahl Pakete Abnahme              4521726    98.84
Doppler 2 Bänder                     55552     1.22
Laufzeit Abnahme                   4574864   100.00
dtypes: float64(7)
```

Table 4.2: Data Description Report: *abnahme*

The *abnahme* file contains 4.574.864 samples with a total of 7 columns. The time frame recorded reaches from 2017-01-11 to 2019-06-30. *NaT* also indicates that there are *Not a Time* values in the dataset, meaning that timestamps are missing. Looking at the amount of missing values in the *abnahme* file, it can be seen that three attributes are missing 100% of the values and another three attributes are above 98%. Only one attribute has a very small number of missing values with only 1.22% missing. In the case of the *Schmierung Tisch 1* attribute, the empirical rule can be applied successfully. What catches the eye is that the *max* and 75 percentile both show a value of 1.0 and the *min*, 25th, and 50th percentile show a value of 0.0. The assumption arises whether this attribute is categorical. Looking at the raw values, a categorical property can be determined. The same applies to the *Doppler 2 Bänder* attribute, which therefore can also be determined as categorical. The *Restfeuchte Platten [BSA]* attribute, on the other hand, shows a jump from 0.174 at the 75th percentile to 2.0 at the *max*. As the *max* value does not fit into the three standard deviations away from the mean, outliers are suspected. Due to a large amount of data in this file, the assumption is pursued that the system did not export the data in quarter-hour steps. Looking at the unique values of the minute characters of the timestamps, this assumption can be confirmed.

```
DatetimeIndex: 162 entries, 2019-01-06 07:29:21 to 2019-12-06 12:50:32
Data columns (total 2 columns):   type             isnull    %null
Artikelnummer / Baugruppe [BSV]*  object                2    1.23%
Ende                              datetime64[ns]        1    0.62%
dtypes: datetime64[ns](1), object(1)

*Knauf Stadtoldendorf.02 Bandstraße.512-Vorbereitungsstation.
Artikelnummer / Baugruppe [BSV]
```

Table 4.3: Data Description Report: *artikelnummer*

The *artikelnummer* file has a very small amount of missing values. It has to be noted that only 162 entries and 2 columns exist, which in comparison to the other files seems quite low. This is further supported by the time frame reaching from 2019-01-06 07:29:21 to 2019-12-06 12:50:32. The reason for this short time frame is that the attributes were introduced to the system at the end of May 2019 as a result of the business understanding phase. What stands out is that even though the records of the other files reach until the end of June 2019, this file stops at the 12th of June. A sensor malfunction has to be expected. One of the business goals is to find relationships between these attributes and the rest of the data. Due to the small time frame, this has to be considered further to not reduce the size of the dataset too much.

```
DatetimeIndex: 98225 entries, 2017-01-11 00:00:00 to NaT
Data columns (total 7 columns):      isnull    %null
Gesamtleistung Brenner [BST]            24    00.02%
Heizwert Erdgas Hu                   98222   100.00%
Laufzeit Trockner [BST]              98225   100.00%
Nassausschuss [BST]                  98225   100.00%
Summe Gasverbrauch Trockner [BST]    98225   100.00%
Summe Stromverbrauch Trockner [BST]  98225   100.00%
Trocknerleistung                        24    00.02%
dtypes: float64(7)
```

Table 4.4: Data Description Report: *trockner*

The *trockner* file contains a total of 98.225 entries and 7 columns. The time frame recorded is the same as for the *abnahme* file. It can be seen that five attributes do not have sufficient to any values and two attributes are only missing 24 values in total. Those two have to be analyzed further with the other ones being considered to be dropped from the dataset. *Gesamtleistung Brenner [BST]*, as well as *Trocknerleistung*, show a jump from the 25th percentile to the 50th percentile with the *min* and 25th percentile value being 0.0. Using the domain knowledge received from the business, the values can be identified as the machine being shutdown or in an idle state. Because the objective is to find relationships of the running machine, dropping the zeros is considered. Furthermore, *Trocknerleistung* shows a jump from the 50th and 75th percentile, to the *max* value. This hints the presence of outliers and has to be addressed further.

```
DatetimeIndex: 944510 entries, 2017-01-10 00:00:00 to NaT
Data columns (total 6 columns):        isnull     %null
spez. therm. Energie pro m^2 [BST]     904031    95.71%
therm. Trockner Effizienz [BST]        904031    95.71%
zu verdampfendes Wasser pro m^2 [BST]  918521    97.25%
Gipsverbrauch aktuell kg/min             9673     1.02%
Plattenbreite in m [BSV]               942278    99.76%
Reinheitsgrad Gips                     944505    99.99%
dtypes: float64(6)
```

Table 4.5: Data Description Report: *trockner_effizienz*

The *trockner_effizienz* is the file containing the two variables representing the energy consumption of the dryer: *spez. therm. Energie pro* m^2 *[BST]* and *therm. Trockner Effizienz [BST]*. It has 944.510 entries and 6 columns. What stands out is a large number of missing values among the target attributes. The recorded time frame indicates a start at 2017-01-10. Domain knowledge and the business understanding

though hint that those attributes were added at the end of May 2019. This means that the system exports the values and their timestamps without taking into account that some attributes did not have values at a specific point in the past. This has to be kept in mind when discussing the time frame recorded. The two target variables have the same amount of missing values, which leads to the assumption that they are recorded at the same time and in the same sample. *Reinheitsgrad Gips* and *Plattenbreite in m [BSV]* do have missing values above 99% of the total data and dropping them is therefore considered. *Gipsverbrauch aktuell kg/min*, on the other hand, shows a high representation among the samples. Continuing with the analysis of the distribution, *spez. therm. Energie pro m^2 [BST]* shows a jump from the 75th percentile to the *max* value. Including the domain knowledge, this again could hint a shutdown or idle state of the machine. The same applies to *Gipsverbrauch aktuell kg/min*. After removing values equal to 0, *spez. therm. Energie pro m^2 [BST]* still shows a jump from the 75 to the *max* value. This again is an indicator for outliers and should be investigated further. Both, *therm. Trockner Effizienz [BST]* and *zu verdampfendes Wasser pro m^2 [BST]* show negative values at the *min* mark. Due to the domain knowledge provided, this is impossible from a physical perspective and values below 0 should be dropped. It makes sense that both attributes show the same anomaly as the business mentioned a connection between the two. What could cause problems when feeding the data into the different machine learning algorithms is the different scale of the target attributes. *spez. therm. Energie pro m^2 [BST]* has a mean of 2.883954 while *therm. Trockner Effizienz [BST]* has a mean of 693.019733. Scaling the output data before the training process is therefore considered a must. Otherwise, e.g. the MAE could be influenced by the different scales.

```
DatetimeIndex: 752586 entries, 2018-01-02 00:00:00 to NaT
Data columns (total 4 columns):    isnull     %null
Öltemperatur                       702348    93.32%
Viskosität                         702346    93.32%
Dielektrizitäts-Konstante              73    00.01%
Leitfähigkeit                      702349    93.32%
dtypes: float64(4)
```

Table 4.6: Data Description Report: *trocknergetriebe*

The *trocknergebtriebe* file has 752.586 entries and 4 columns. The time frame recorded starts at 2018-01-02. Using the insights gathered in the preceding section and looking at the number of missing values, the assumption lies near that the record of *Dielektrizitäts-Konstante* started at an earlier point in time. This is further supported by the similar number of missing values of the other attributes. Distribution-wise only *Viskosität* and *Leitfähigkeit* hint potential outliers. Because of the insignificance of the attributes, outlier detection is not considered, as it would further reduce the amount of data samples.

DatetimeIndex: 392803 entries, 2017-01-11 00:00:00 to NaT		
Data columns (total 8 columns):	isnull	%null
Laufzeit Mischer [BSV]	333872	85.00%
Laufmeter Bandstraße	392803	100.00%
Summe Stromverbrauch Vorbereitungssation	392803	100.00%
Plattenbreite [BSV]	392368	99.89%
Bandgeschwindigkeit ABB1 [BSV]	294210	74.90%
Artikelnummer / Baugruppe [BSV]	392308	99.87%
Bandgeschwindigkeit ABB2 [BSV]	334293	85.10%
Bandgeschwindigkeit ABB1 Soll [BSV]	122115	31.09%
dtypes: float64(4)		

Table 4.7: Data Description Report: *vorbereitungsstation*

The *vorbereitungsstation* contains 392.803 entries and 8 columns. The time frame reaches from 2017-01-11 to 2019-06-30 and missing timestamps are present as indicated by *NaT*. Two attributes have 100% missing values and another one is close with 99.89%. Those are considered to be dropped. Utilizing domain knowledge, *Bandgeschwindigkeit ABB1 Soll [BSV]* can be classified as an attribute representing a machine setting. This is indicated by the keyword *Soll*. Attributes containing this keyword are only recorded when the setting is changed. The businesses proposed dropping all of those attributes because it is said that the machine can regulate very accurate according to its settings. *Bandgeschwindigkeit ABB1 [BSV]* and *Bandgeschwindigkeit ABB2 [BSV]* both show a jump from the 25th percentile to the 50th percentile. As those attributes represent the speed that the assembly line is running at, zeros indicate an idle or shutdown state and are therefore dropped. For *Laufzeit Mischer [BSV]* on the other hand, a jump from the 75th percentile to the *max* value is recorded, indicating outliers. The attribute name *Artikelnummer / Baugruppe [BSV]* hints a connection to the *Knauf Stadtoldendorf.02 Bandstraße.512-Vorbereitungsstation.Artikelnummer / Baugruppe [BSV]* attribute of the *artikelnummer* file. Joining them on their index and dropping *NaN* (Not a Number) or missing values would return rows that include a value for both attributes. This way a matching could be identified. Because no row is returned, no connection between the two attributes can be identified.

DatetimeIndex: 641605 entries, 2017-01-11 00:00:00 to NaT		
Data columns (total 14 columns):	isnull	%null
Klappenstellung Abluft Zone 02 [BST]	641519	99.98%
Klappenstellung Abluft Zone 01 zu WRG SOLL [BST]	641220	99.94%
Klappenstellung Abluft Zone 01 zu WRG [BST]	639496	99.67%
Klappenstellung Abluft Zone 02 SOLL [BST]	641242	99.94%
Unterdruck Zone 1 Tür 2 [BST]	59406	9.26%
Klappenstellung Abluft Zone 02 zu WRG SOLL [BST]	640491	99.83%
Klappenstellung Abluft Zone 02 zu WRG [BST]	640625	99.85%
Differenzdruck WT Frischluft [BST]	526680	82.09%
Ablufttemperatur vor WRG [BST]	543009	84.63%
Ablufttemperatur nach WRG [BST]	543009	84.63%
Frischlufttemperatur nach WRG [BST]	543009	84.63%
Klappenstellung Abluft Zone 01 SOLL [BST]	639811	99.72%
Klappenstellung Abluft Zone 01 [BST]	631374	98.41%
Frischlufttemperatur vor WRG [BST]	543126	84.65%
dtypes: float64(14)		

Table 4.8: Data Description Report: *waerme_rueckgewinnung*

Many of the 14 columns of the *waerme_rueckgewinnung* file show a high percentage of missing values. The total number of samples is 641.605 with a time frame recorded starting at 2017-01-11 and containing missing values. It also has to be noted that there are many attributes containing the keyword *SOLL*. As discussed earlier, those have to be dropped. What stands out, are attributes containing the attribute names of those without the keyword. These are the actual measurements following the change of settings. They are only recorded when a change occurs, explaining the small number of values. Due to this, the values have to be propagated forward among the entries until another change is recorded. Besides those findings, *Unterdruck Zone 1 Tür 2 [BST]* is only missing 9.26% of the total number of samples. The attributes *Differenzdruck WT Frischluft [BST]*, *Ablufttemperatur vor WRG [BST]*, *Ablufttemperatur nach WRG [BST]*, *Frischlufttemperatur nach WRG [BST]* and *Frischlufttemperatur vor WRG [BST]* that by domain knowledge contain temperatures of the air, share a similar value population. The assumption lies near that the values are always recorded together at the same time. Looking at the distributions, *Klappenstellung Abluft Zone 02 [BST]* shows a mean and standard deviation of 0.0 with every percentile being 0.0. This means that all values are 0.0 which does not contain sufficient information for further use and therefore is considered for dropping. The other attributes show significant jumps from the 75th percentile to the *max* which has to be investigated further.

DatetimeIndex: 895698 entries, 2017-01-11 00:00:00 to NaT		
Data columns (total 9 columns):	isnull	%null
Regel Temperatur Zone 01 SOLL [BST]	887955	99.14%
Regel Temperatur Zone 01 [BST]	797103	88.99%
Temperatur Zone 01 Einblasseite [BST]	797106	88.99%
Temperatur Zone 01 Ausblasseite [BST]	797105	88.99%
Mittlere Temperatur Zone 01 [BST]	7689	00.86%
Feuchtebeladung Luft Zone 01 [BST]	797132	89.00%
Brennerleistung Zone 01 [BST]	797102	88.99%
Frequenz Umluftventilator re. Zone 01 [BST]	799184	89.22%
Frequenz Umluftventilator li. Zone 01 [BST]	799184	89.22%
dtypes: float64(9)		

Table 4.9: Data Description Report: *zone01*

The file *zone01* has 895.698 entries with 9 columns and the time frame recorded reaches from 2017-01-11 to 2019-06-30 and has missing values. Analyzing the amount of those, the assumption can be made that *Mittlere Temperatur Zone 01 [BST]* is either recorded at a different time interval or records exist before the other sensors were added to the system. The other attributes show a similar amount of values missing, supporting this assumption. Again, one attribute contains the keyword *SOLL* and is considered for dropping. Business insights also hint that both *Frequenz Umluftventilator* attributes represent the frequency of fans which always operate at the same frequency as each other. After investigating this further, dropping one of those can be taken into consideration. Looking at the value distribution of the attributes, it becomes clear that all attributes show significant jumps from the 25th to the 50th percentile. This can be explained using domain knowledge. The dryer is split into zones that the plasterboards have to pass during the drying process. If the machine is in an idle or shutdown state, no plasterboards move through the dryer and no additional air is heated up, hence the sensors measuring the temperature or board-related metrics return small values. The connection between the *Frequenz Umluftventilator* attributes is further supported by their nearly identical mean, standard deviation, count and percentiles.

```
DatetimeIndex: 111329 entries, 2017-01-11 00:00:00 to NaT
Data columns (total 10 columns):                isnull     %null
Regel Temperatur Zone 02 [BST]                   12737     11.44%
Plattenfeuchte Zone 02 [BST]                     12737     11.44%
Regel Temperatur Zone 02 SOLL [BST]             102237     91.83%
Temperatur Zone 02 Einblasseite [BST]            12737     11.44%
Temperatur Zone 02 Ausblasseite [BST]            12737     11.44%
Feuchtebeladung Luft Zone 02 [BST]               12737     11.44%
Brennerleistung Zone 02 [BST]                    12737     11.44%
Frequenz Umluftventilator li. Zone 02 [BST]      10453      9.39%
Mittlere Temperatur Zone 02 [BST]                12737     11.44%
Frequenz Umluftventilator re. Zone 02 [BST]      14118     12.68%
dtypes: float64(10)
```

Table 4.10: Data Description Report: *zone02*

Even though the analysis of *zone01* implies that there is a connection between the zones, *zone02* has a significantly lower amount of entries than *zone01* with only 111.329. It has similar attributes though, except *Plattenfeuchte Zone 02 [BST]* which is not present in *zone01*. The time frame recorded is also the same, leaving only the explanation of *zone01* recording the data in a different time interval. Nonetheless, the assumption that most of the attributes are recorded at the same time is further supported by the file, who's seven attributes have the exact same amount of values. Overall the attribute with keyword *SOLL* is the only attribute with a large number of missing values. Like the preceding file, the distributions of the attributes are very similar with significant jumps from the 25th to the 50th percentile due to the reasons explained in the preceding paragraph.

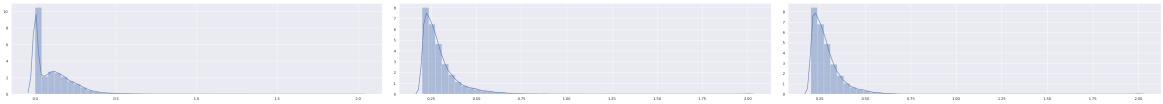
The files *zone03* and *zone04* show the same findings as the *zone02* file with a quite similar amount of entries, missing values, and distributions. The columns of *zone03* mirror the ones of *zone02*. File *zone04* differs by not having the *Feuchtebeladung Luft* attribute. There is no explanation for this, but a guess can be made that the engineer connecting the sensors did simply forget to add this specific one to the system.

To sum up the report, it can be said that all files are recorded in a similar time frame. Some exceptions and anomalies hint that this could be misinformation due to the systems export behavior. Every file has an attribute of type *datetime64[ns]* which is used as an index for later merging the files together. Problems could arise because of the partly large amount of missing values, especially among the target variables. Therefore, all data is used to gain insights about relationships among the attributes and only the data connected to the target variables by its index is used for training. Another issue that has to be investigated further is the unbalanced distribution of most attributes. To find solutions to those problems and to gain an even deeper insight into the data, EDA is performed next.

Exploratory Data Analysis

By definition, the EDA also includes the use of pure statistical techniques to analyze the data. Because they were already discussed in the previous chapter and because literature puts great focus on visualization, pure statistical techniques are left out in this section. For the visualizations, the python library *seaborn* is used. It is based on matplotlib and "[...] provides a high-level interface for drawing attractive and informative statistical graphics."[29] Again each file is analyzed, focusing on the anomalies detected in the data description and exploration report.

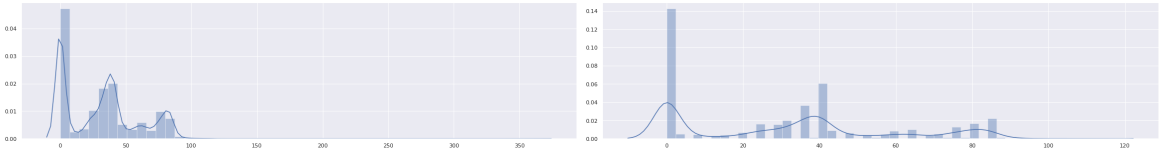
In the previous chapter, the *Restfeuchte Platten [BSA]* attribute of the *abnahme* file shows signs of outliers. The visualization supports this assumption with a large number of values close or equal to 0 and the distribution being drawn towards those. Only very few values can be identified towards the maximum value. The business provides information that the values have to be bigger than 0.2 to be representative of reality. To improve the representability, the attribute is cropped to fit this range. To validate the presence of outliers, outlier detection is performed and the outliers are dropped from the dataset. Visualizing the new data, it can be seen that cropping the attribute has a positive impact on the distribution. The outlier detection algorithm though did not detect enough outliers to improve the distribution. It is therefore not considered for the data preparation step, as removing detected outliers does not make a significant difference but reducing the size of the dataset.



(a) initial distribution (b) cropped $0.2 < x$ (c) cropped, without outliers

Figure 4.1: *Restfeuchte Platten [BSA]*

Anzahl Pakete Abnahme also indicated outliers. In this case, outlier detection successfully detects those. The distribution indicates that there are still outliers close to zero to be expected. Because this assumption can not be verified, the values are kept. Nonetheless, significant improvement in terms of maximum values is achieved by only dropping 176 samples.



(a) initial distribution (b) without outliers

Figure 4.2: *Anzahl Pakete Abnahme*

As mentioned before, *artikelnummer* includes the names of the board currently in production. The business mentioned that there are four main products and focus should lie on those. To verify this information, the occurrence of each unique product name is plotted in 4.3. It can be seen that there are indeed four products produced most with counts of 31, 26 and 16. Looking at the bigger scope, this finding can not be considered representative with another three products around the range of 10. Because of the small size of the dataset of only 162 samples, the distribution could easily change when more data is recorded. Because the values only represent the beginning of a production cycle of a specific product, they have to be propagated, filling in empty values of samples falling in between the time span of production. Inspecting the string values, also similarities of the different product names can be identified. After consulting the business about this, information is provided confirming the findings and recommending splitting the strings into its components. The components are divided by the '-' character and start with the product type, followed by the height, width and length of the board and the type of edge produced. The third and fifth are not considered influential by the business. This leads to only using the first, second and fourth components for further analysis. To confirm the four most produced products, the product types are plotted in 4.4. Indeed, the four products found in the 4.3 are represented in the top three product types. Production time starts at the timestamp of the index and ends at the time stamp of the *Ende* attribute.

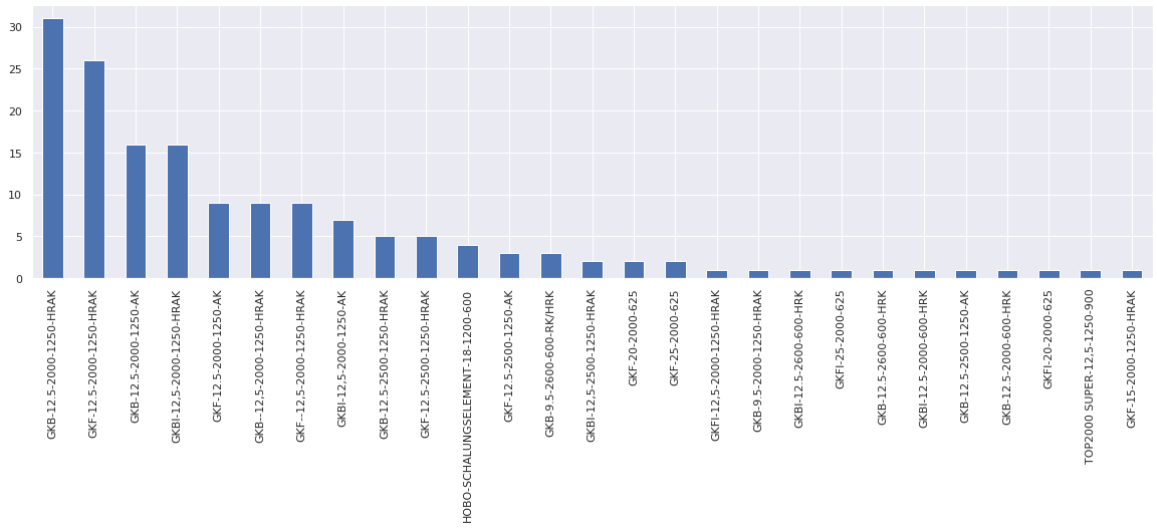


Figure 4.3: *Artikelnummer / Baugruppe [BSV]*

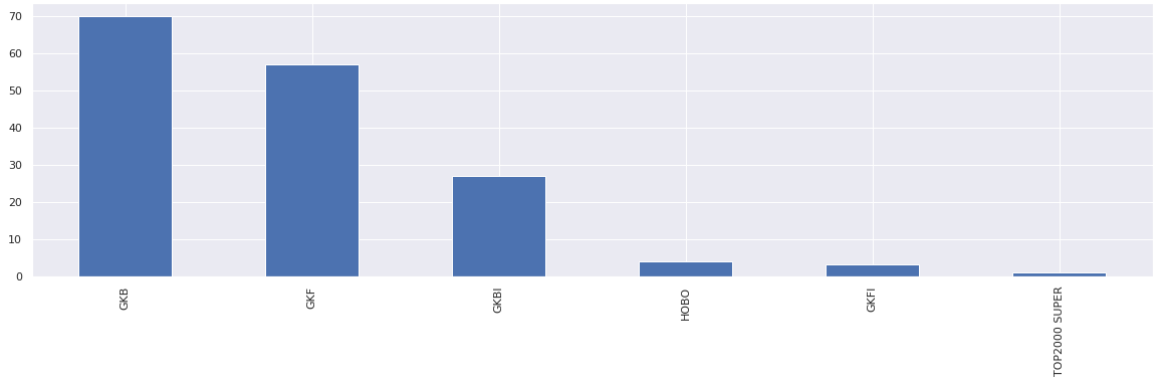
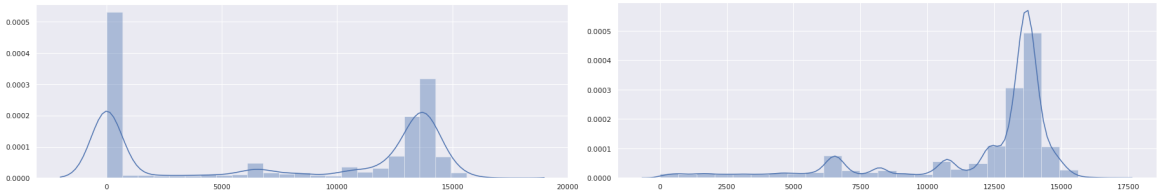


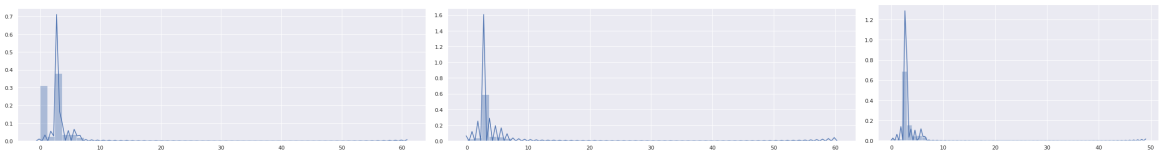
Figure 4.4: Product Types

The attributes of *Gesamtleistung Brenner [BST]* and *Trocknerleistung* of the *trockner* file showed many zeros which were identified as the machine is in an idle or shutdown state. As it was decided to drop those, the decision must be verified further. The visualization shows drastic improvement towards a distribution without outliers for the *Gesamtleistung Brenner [BST]* attribute. To also improve *Trocknerleistung*, outlier detection is used. It can be seen that some of the values close to the maximum value were dropped by the practice, leading to a better but not perfect result.



(a) initial distribution (b) without zeros

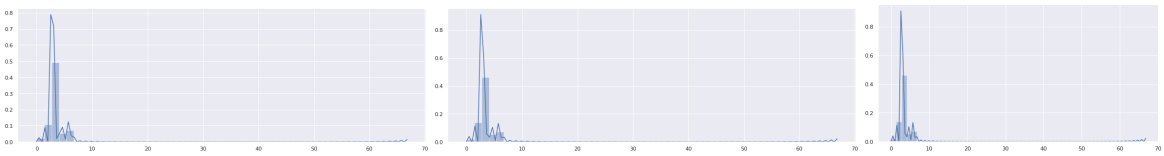
Figure 4.5: *Gesamtleistung Brenner [BST]*



(a) initial distribution (b) without zeros (c) without outliers & zeros

Figure 4.6: *Trocknerleistung*

The next file to be analyzed is *trockner_effizienz*. This file includes the target attributes, which should be as close to a normal distribution as possible to ensure high performance of the machine learning algorithms, even if those are not sensitive to outliers. Again, zeros are dropped due to the machine's shutdown or idle state. As this does not achieve a sufficient improvement on both attributes, outlier detection is performed on *spez. therm. Energie pro* m^2 *[BST]* dropping 7129 samples. The visualization shows that there are still potential outliers bigger than 10.0. Because removing them would lead to a reduction of the samples and the outliers can not be confirmed, cropping them is not considered.

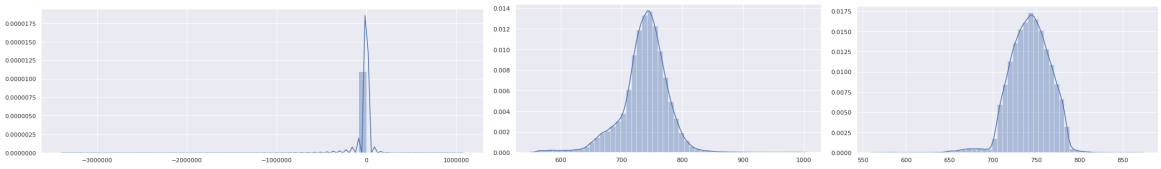


(a) initial distribution (b) without zeros (c) without outliers & zeros

Figure 4.7: *spez. therm. Energie pro* m^2 *[BST]*

The distribution plot of *therm. Trockner Effizienz [BST]* shows values far beyond 0 reaching into the negative space. As discussed earlier, this is physically impossible

and a sensor malfunction or calculation error is suspected. By discussing this issue with a plant engineer, it becomes clear that the value physically can only be between 560 (minimum energy the machine can run on) and 1000 (worst value that can be achieved). After cropping the dataset accordingly, a distribution very close to a normal distribution is reached. This is further improved by outlier detection dropping 6357 values. Comparing both attributes on a line plot, a negative correlation can be assumed. To confirm this assumption, a correlation matrix is created, showing the correlation amongst attributes with 1 representing the highest positive and -1 representing the highest negative correlation. With a value of -0.205, the assumption can be confirmed to an extent, but the correlation is very low due to the value being closer to 0 than to 1.



(a) initial distribution (b) cropped $560 < x < 1000$ (c) cropped, without outliers

Figure 4.8: *therm. Trockner Effizienz [BST]*

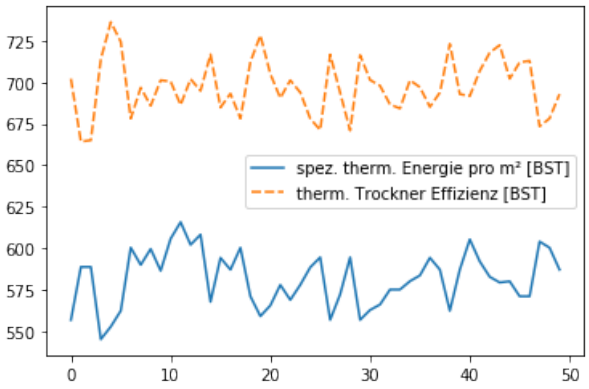
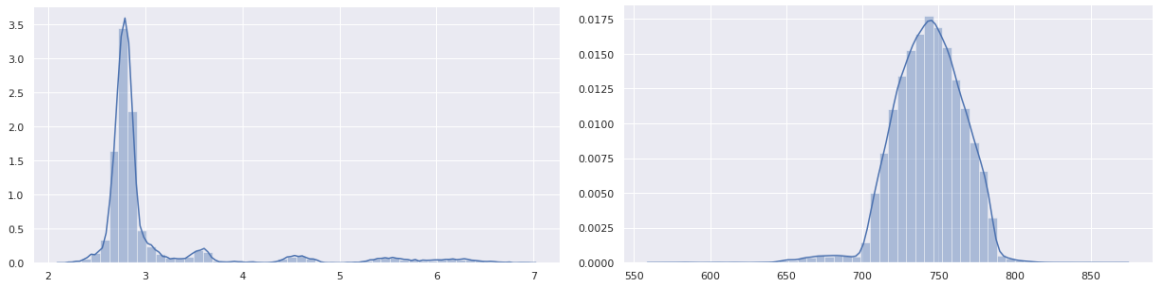


Figure 4.9: *spez. therm. Energie pro m^2 [BST] & therm. Trockner Effizienz [BST]* (scaled)

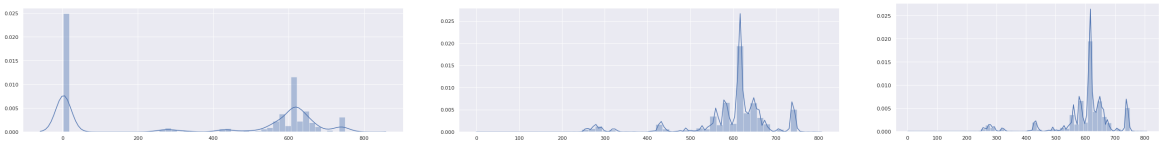
Because both values represent the energy consumption, the two attributes are joined on their index and the missing values are dropped. When now plotting both distributions in 4.10 it can be seen that the distribution of the *spez. therm. Energie pro m^2 [BST]* attribute improved even further. The correlation of the merged attributes is reduced to -0.244. This still represents a very low correlation which in regards to the attributes being the target values is considered positive as they do not contain much redundant information.



(a) *spez. therm. Energie pro* m^2 *[BST]* (b) *therm. Trockner Effizienz [BST]*

Figure 4.10: *therm. Trockner Effizienz [BST]*

Another attribute with the issue of being connected to the machine state is *'Gipsverbrauch aktuell kg/min'*. After dropping the zeros without making a noticeable difference, it is decided that 10.0 is the threshold of choice. Indeed after cropping the data, the distribution improves. Further progress is made by applying outlier detection and dropping 391 outliers.



(a) initial distribution (b) cropped x < 10 (c) cropped, without outliers

Figure 4.11: *'Gipsverbrauch aktuell kg/min'*

When visualizing the *trocknergetriebe* attributes, the distributions can be identified as uneven. Even after performing outlier detection overall attributes at once and dropping 6987 outliers, no improvement is achieved. Because the attributes are not considered important, further steps are not taken and outlier detection is not considered for data preparation. Also, after discussing these issues with the business, a connection between the attributes and the target is considered highly unlikely and dropping them is proposed. The same observations are made for the *vorbereitungsstation* file. Though the attributes *Bandgeschwindigkeit ABB1 [BSV]* and *Bandgeschwindigkeit ABB2 [BSV]* can be improved by dropping the zeros with their relation to the machine state.

The histograms of the *waerme_rueckgewinnung* file show similar distributions for *Ablufttemperatur vor WRG [BST]*, *Ablufttemperatur nach WRG [BST]* and *Frischluft-temperatur nach WRG [BST]*. A reason for this could be a high correlation amongst

them. Inspecting the correlation matrix of the dataframe, a correlation higher than 0.97 can be identified for each attribute relationship. This can be interpreted as an information redundancy amongst those. To reduce dimensionality, two of the three attributes can be dropped without losing too much information.

Looking at the line plot of *zone01*'s temperature attributes, it can be seen that all attributes besides *Temperatur Zone 01 Einblasseite [BST]* and *Regel Temperatur Zone 01 [BST]* follow each other and are therefore assumed to share a high correlation. Creating a correlation matrix, these observations can be refuted, as all values correlate with a value of 0.94 and higher. The attribute containing the keyword *SOLL* shows a very low correlation with scores between 0.15 and 0.36. This does not raise a problem because the attributes containing the keyword are dropped from the datasets. The high correlation among the other attributes allows for dropping all but one, as the amount of information lost is low in comparison to the benefits of dimensionality reduction. *Mittlere Temperatur Zone 01 [BST]* is chosen as it has the highest correlation values with the other attributes.

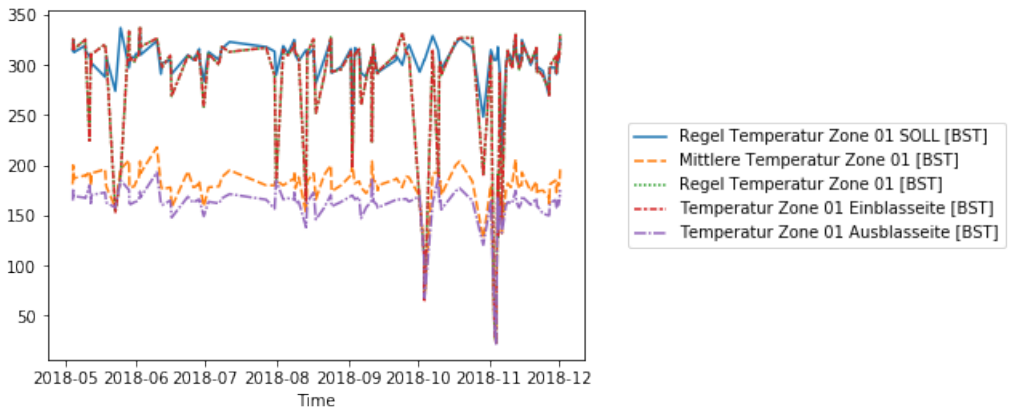
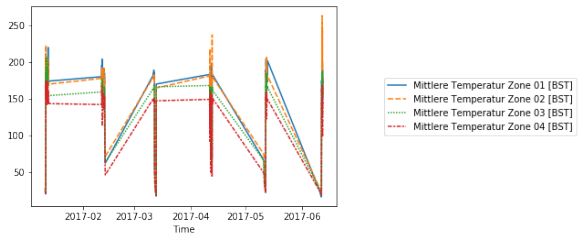


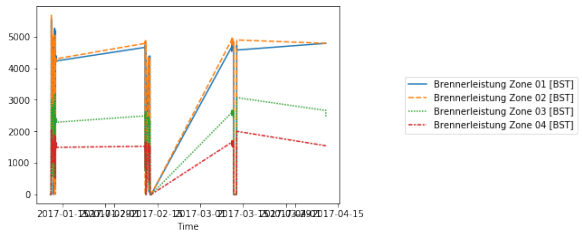
Figure 4.12: *zone01* Temperature Attribute Comparison

Because the data description and exploration report both assume that the files *zone01*, *zone02*, *zone03* and *zone04* share similarities, the same relationships among the attributes are also assumed for them. This is confirmed by plotting the selected attribute of *zone01* and the other files in a line plot. Again, the patterns of all attributes are very similar, leading to the assumption that a high correlation exists. The correlation values confirm this with values of 0.96 and higher. Therefore also only a single attribute is selected. This observation can also be found when comparing the *Brennerleistung* attributes of the different files leading to the same result. Because *zone01* does not contain a *Plattenfeuchte* attribute, the other files are compared separately. Supported by a line plot and a correlation matrix, no high correlation amongst them can be found and all attributes have to be used. Another anomaly that was detected is the jump

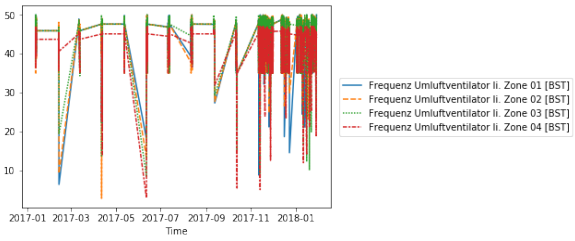
from the 25th to 50th percentile of the *Brennerleistung* attributes. Dropping all values lower or equal to zero does improve the distribution, but loses 63309 samples. Due to the connection to the machine state, this must be accepted. Assumptions made about the two *Frequenz Umluftventilator* attributes of each file can also be confirmed by visualizing them in a histogram. Both attributes show an identical distribution leading to only one being used for further analysis. The zeros are not considered to be dropped as the business provides information about both fans being able to be shut down without the machine being in a shutdown or idle state. Comparing those attributes of the different files, a high correlation is expected and later confirmed with correlation values of 0.80 and higher. These findings allow the use of only one attribute to represent the information.



(a) *Mittlere Temperatur*



(b) *Brennerleistung*



(c) *Frequenz Umluftventilator li.*

Figure 4.13: Attribute Comparison *zone01*, *zone02*, *zone03* & *zone04*

To sum up the EDA, the anomalies discovered in the data description and exploration report can be resolved for the most part by carefully analyzing the visualization of the data. Cropping and removing outliers shows potential in improving the distribution. Especially for the target values, the results are significantly better than the initial data. Even though the objectives defined by the business regarding the target values can be answered using the data, a correlation with the product produced seems to be difficult as the amount of data provided is too small. Additionally, valuable assumption can be made to support and improve the preparation and modeling process. The decisions developed for preparing each attribute are listed in A.3.1.

4.3 Data Preparation

4.3.1 Data Cleaning

The steps of the data preparation are carefully ordered by the amount of potential data dropped and the amount of processing power required. By performing tasks of high processing power last and tasks that potentially drop a lot of data first, the execution time can be reduced. The knowledge gained from the data analysis is used to perform a feature selection. Columns that do not provide enough values or can be ignored due to domain knowledge are dropped from the dataset. Because dropping columns does not require a lot of processing power and potentially reduces the size of the dataset, it is performed first.

Because further steps can only be performed if no missing values are present, taking care of those is done next. As seen in the data analysis and incorporating the domain knowledge, attributes were identified that record changes in a setting or are not recorded as regularly as others. Due to this, dropping every row containing a missing value is not a valid solution. The values are therefore propagated and the subsequent samples are filled with the values until a new timestamp indicating the next change in settings is recorded. Before propagating, the data is sorted by index to ensure that the entries are ordered by their recorded timestamp. Otherwise, values would be allocated that do not match the state of the sample.

The conversation with the business showed that some values do not represent reality sufficiently and contain inaccuracies or are erroneous. This is especially important when it comes to the target attributes. By cropping those values, outliers that were caused by sensor malfunction or erroneous calculations are removed and the distribution is improved for training. The range for cropping is defined by utilizing the domain knowledge.

Like mentioned earlier, the business proposed splitting the attribute containing the product names into its components. Additionally, it is suggested to drop the ones that are not expected to have a high impact on the target values based on physical relationships. The different components are incorporated into the string representing the product name and separated by the '-' character. By splitting them, the individual components are extracted from the original attribute. Because new features are generated incorporating domain knowledge, this is an example of feature engineering even though the process is an inversion of the one described in 3.5.2.

Some features are identified as categorical in the preceding analysis which also applies to the features engineered in the preceding step. Because the range difference in those can affect some of the machine learning algorithms in a negative way, they are encoded into integers increasing from 0 in steps of 1.

On the gypsum boardline, the sensors are placed at different positions. Those positions are indicated by including the keywords *[BSV]*, *[BSA]* and *[BST]* into the attribute name. Graphic 3.1 shows how the machine is separated. The values at *[BSV]* are recorded 10 minutes in advance to the *[BST]* values. The *[BSA]* values, on the other hand, are recorded 25 minutes later than the *[BST]* values. To get the values belonging to each other, the timestamps are updated accordingly. As most of the values belong to *[BST]*, the other sections are mapped to it accordingly.

Because the data analysis showed that the ResMa did not export the data in quarter-hour steps, the timestamps are aggregated to fulfill this prerequisite. If the aggregation has multiple values for an attribute, the mean is taken.

4.3.2 Outlier Detection

In the data analysis it can be seen that performing outlier detection on some attributes can improve the distribution. To detect outliers, scikit-learn provides different approaches that can be found in the documentation [30]. Out of those *LocalOutlierFactor*, *IsolationForest*, and *EllipticEnvelope* are chosen as *OneClassSVM* is not recommended for outliers but for novelty detection.[cf. 31] A more detailed description of the algorithms used by those classes is not given as they are not considered of high importance to the project or its understanding.

To get a more accurate result and to decrease the possibility of outliers even further, all three detection algorithms are used. Subsequently, these results are combined and the samples classified as outliers are dropped from the dataset. It also has to be mentioned that even though the algorithms can deal with multiple attributes, they are only applied to a single attribute at once.

4.3.3 Data Joining

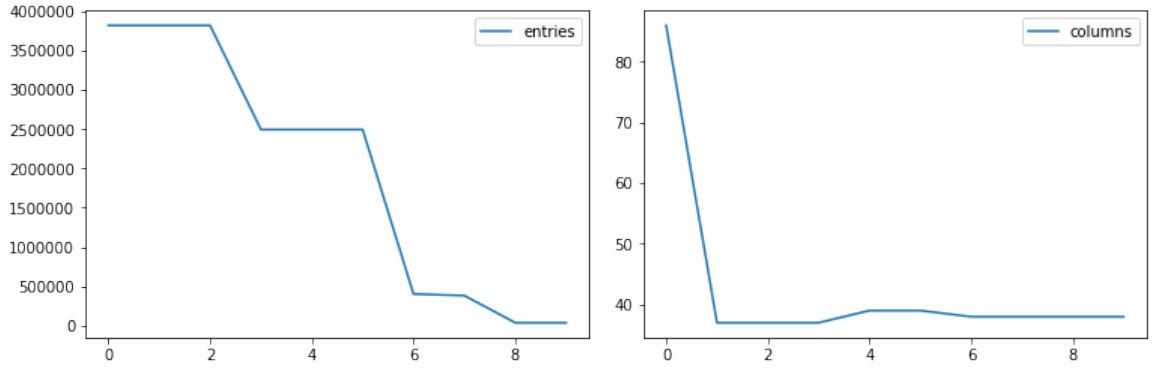
After the data is cleaned and the outliers are detected and removed, the data of the different files are joined on the DateTime index. If two or more entries share the same index, the mean of the values is used for aggregation.

Because so far, the attributes of the *artikelnummer* file were only processed separately, they are now assigned to the samples falling into the production range of the products. This is done by first propagating the engineered product name components and the *Ende* attribute, marking the end of a product produced. To now identify the samples for which a product is recorded, the timestamps are compared. Because the samples are still ordered by their DateTime index, a match is made every time the DateTime of the index is before the DateTime of the propagated *Ende* attribute. If no match

is made, the values of engineered attributes are dropped. Furthermore, the *Ende* attribute is dropped as the information is no longer required.

Joining the data leads to missing values because not every attribute has a recording for each available timestamp. A solution has to be found to deal with those, as most machine learning algorithms can not deal with such. To solve this issue, scikit-learn provides different impute strategies that generate the missing values out of the given ones. The two most common classes used are *SimpleImputer* and *IterativeImputer*. The first is using different strategies like 'mean', 'median', 'most_frequent' or 'constant' to fill the missing values.[cf. 32] The second uses a more sophisticated approach, modeling "[...] each feature with missing values as a function of other features, and uses that estimate for imputation."[33] To influence the modeling part, different estimators can be passed to the class.[cf. 34] In this project, the default option *BayesianRidge* is used. Even though the estimator is still experimental, it is used for the project due to its promising results. Because the underlying algorithms are not crucial for this project, it is referred to the scikit-learn documentation of the classes instead.[cf. 32, cf. 34] Scaling is another important part of the data preprocessing and is therefore also performed on the data. The presence of outliers can still not be excluded, because detection was not performed on every single attribute. Due to this, standardization is used as the type of scaling.

After applying all of the techniques mentioned above, the output data went from the initial 3.821.441 entries and 86 columns to 36820 entries and 38 columns. The progression in regards to the different steps is visualized in 4.14. It can be seen that a focus lies on dropping the columns as early as possible and the single entries at the end. This leads to a steeper curve in dataset size at the beginning, directly benefiting the required processing power and memory. Due to the different actions performed, no missing values are present in the output dataset and outliers are removed to an extent.



(a) Entry Loss (b) Column Loss

Figure 4.14: Data Preprocessing Loss

4.4 Modeling

4.4.1 Training

The training process starts by fetching the training data, shuffling and splitting it into training and test data using the *train_test_split*-method of scikit-learn. Next, an instance of the model is created and a configuration can be made by setting the parameters.

Keras does not support a scikit-learn integration by default. By using the wrapper this functionality can be added. Setting up the estimator first requires a function creating the Keras model. In this case, a sequential model is chosen with input dimension equal to the number of features and the output dimension of size 2 due to the number of targets. Additionally, a hidden layer is added which allows a dynamic number of hidden layer neurons to be set. For the activation function of the hidden layer, ReLU is chosen as it is fast to compute and recommended for hidden layers. The output layer uses a linear activation function as the output is supposed to be of continuous type. Because also the MAE is measured which directly depends on the scale of the output values, the targets are scaled to achieve equal influence on the metric. This would not be the case without scaling, because the estimation of a value with a bigger mean is usually more likely to differ from the target by a higher value leading to a bigger impact on the MAE. Furthermore, the optimizer is set using a variable for later optimization. A function returning the model is then passed to the wrapper which then can be used as a native scikit-learn model. The resulting model can therefore also be used for a randomized search with cross validation to tune the hyperparameters.

To combine the *PolynomialFeatures* class, creating the polynomial features, with a model, the *Pipeline* class can be used. A pipeline object allows to "[sequentially] apply a list of transforms and a final estimator [...]", meaning that the polynomial features can be integrated into the process and is applied before the model is trained or predictions are made.[35]

4.4.2 Hyperparameter Tuning

Because of the small dataset, it is decided the best course of action is to perform a cross validation to guard against overfitting. Additionally, grid search is performed to find the best possible parameter values to improve the model. Because many different models were chosen and the training of every single parameter combination would lead to a large amount of time needed, a randomized search is preferred instead. Scikit-learn provides this functionality with its *RandomizedSearchCV* class, combining both randomized search and cross validation. The randomized search is set to 10 iterations,

using a 5 fold cross validation and taking the R^2 as its scoring function to improve the models. A best practice is to use the training set for the cross validation and then validate the trained model with the training set. Even though the preprocessing does not result in a large amount of data, this practice is applied. The small dataset is already prone to overfitting and by following this approach the possibility of overfitting can be reduced. For each estimator type, a parameter grid is defined that is later used by the randomized search to sample different combinations and to train the models on those. The parameter grids for each estimator type can be found in the appendix under A.4.3. The grids are created by selecting the parameters provided by the scikit-learn documentation that make a potential impact on the model performance and choosing values around their defaults.

The classes used for each specific model are mentioned in A.4.2. This information is required to interpret the best parameter combinations found.

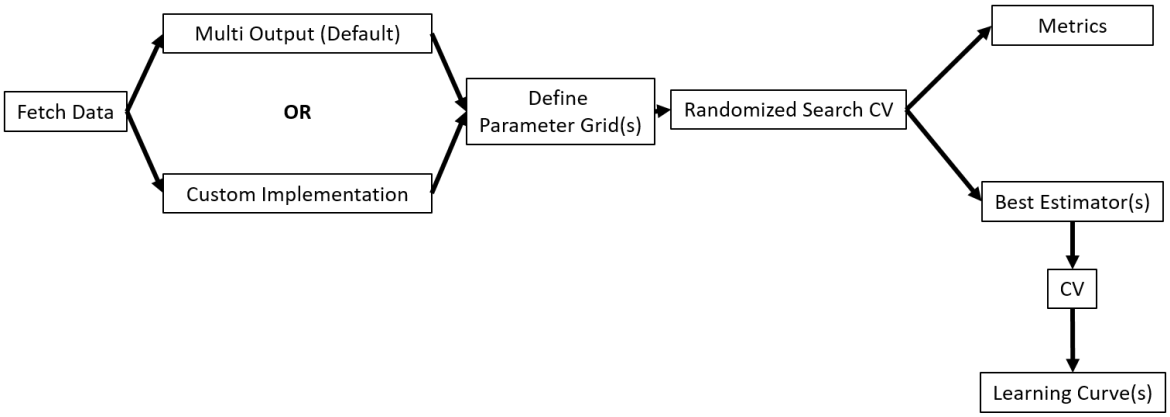


Figure 4.15: Modeling Process

4.4.3 Evaluation

	MSE		MAE		R2	
Model	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
XGB	0.0023	0.0038	0.0249	0.0288	0.9977	0.9962
RandomForest	0.0026	0.0054	0.0272	0.0347	0.9973	0.9948
ANN	0.0280	0.0346	0.0422	0.0426	0.9718	0.9600
SVR	0.0439	0.0736	0.0229	0.0381	0.9557	0.9311
Polynomial Ridge	0.0742	0.1147	0.0997	0.1082	0.9255	0.8868
Polynomial LinearRegression	0.0711	0.1222	0.0969	0.1067	0.9286	0.8793
LinearRegression	0.1771	0.1684	0.1200	0.1189	0.8222	0.8337
Ridge	0.1773	0.1690	0.1211	0.1201	0.8220	0.8331
Lasso	0.1775	0.1693	0.1215	0.1206	0.8218	0.8328
ElasticNet	0.1823	0.1748	0.1271	0.1261	0.8170	0.8275

Table 4.11: Model Scores and Errors

Linear Regression

After the components are defined, the models are trained and predictions made. To evaluate the resulting models, the MSE, MAE, R^2 score and the best estimator found by the randomized search are determined. Additionally, a cross validation using the *KFold* class with 5 folds is used to plot the learning curve of the best estimator over the course of training. A summary of the procedure is given in 4.15. In the following discussion the term linear regression refers to the implementation of the *LinearRegression* class and not to the term defined in 2.3.2 as this would also include the regularized models.

After the data is prepared, the linear model is trained. By default, the implementation supports multi-output regression. Against the assumption that the data can not be represented by a linear model, low error rates are reached for both the training and the test set. The R^2 scores of 0.8222 and 0.8337 are close to 1, supporting this observation. This is uncommon as the model usually performs better on the training set. A reason for this could be the cross validation performed during the randomized search. This way, the training data is not overfitted, reducing bias and increasing variance.

A learning curve is created to visualize the R^2 score development in regards to the number of training samples used. This is done by using the best parameter combination found and training a model with those using cross validation. During the training, both the training and the cross validation score are recorded and later plotted. As expected, the training score starts above the cross validation score and gets closer to it as the number of samples increases. A cross validation score close to the training

score usually indicates a model that generalizes well on new data with respect to the training performance. It has to be kept in mind that overfitting is still possible if the amount of data used is not of sufficient sample size. The lower the amount of data, the lower the number of patterns that are represented in it.

Looking at the best parameter combination found, it can be seen that the *fit_intercept* parameter is set to false. This means that no intercept is used in the calculations made for training the model.[cf. 36] The scikit-learn documentation states that if this parameter is set to false, the *normalize* parameter is ignored.[cf. 36]

Linear Regression with Regularization

Ridge regression shows similar results for the metrics as the linear regression. Because of this, the α value determining the amount of regularization used is expected to be close or equal to 0. If this is the case, ridge regression would be equal to linear regression. Looking at the best parameter combination found, it can be seen that *fit_intercept* is again set to false, mirroring the behavior of the linear regression. The value for α is 1e-08 which is indeed close to 0. It is also the value closest to 0 provided by the parameter grid. Validating on the test set, an MSE of 0.1690 and an MAE of 0.1201 are measured with an R^2 of 0.8331. The metrics regarding the test set are considered more meaningful as they represent the generalization capability of the model. Those findings are further supported by the learning curve also being very similar to the linear regression.

Lasso regression also applies regularization with parameter α determining the amount used. Examining the best parameters returned by the randomized search, it can be seen that again the lowest α values of 1e-08 performed best. For *fit_intercept* again false is chosen by the algorithm. Even though not much regularization is applied, the metric scores are slightly worse than the ones of the preceding models. For the test set, an MSE of 0.1693 and an MAE of 0.1215 is achieved with an R^2 of 0.8238. The metrics of the training set are again slightly worse than the ones of the test set. Because the regularization leads to a model performance worse than the one of the linear model, regularization does not seem like a good approach to improve the model.

The elastic net achieves similar but slightly worse scores than the preceding regularized models. The results on the training set are an MSE of 0.1748 and an MAE of 0.1261 with an R^2 score of 0.8275. Even though improvements are gained from the training to the test set, the scores are still not as good as the ones of the linear model without regularization. The parameter *l1_ratio* represents the r value of 2.10, determining the mix of the two norms. Inspecting the best parameter combination, it can be seen that it is set to 1, making the algorithm equivalent to lasso regression. Comparing the parameters to the lasso regression, it becomes clear that the same hyperparameters

are chosen for elastic net. The only difference, which is also expected to be responsible for the different scores is the *tol* parameter which is set to 0.0001 instead of 1e-08. This parameter is used as stopping criterion, terminating the algorithm when reached. Because the lower value is chosen, the training process is expected to have stopped earlier. Stopping early usually is used to stop the training if no significant improvement is made from one iteration to another. This way, overfitting the data is prevented. The learning curve is similar to the ones of ridge and lasso regression, but the cross validation score converges faster to the training score. This means that in comparison, fewer data samples are required to achieve a similar model performance.

Polynomial Regression

For the polynomial feature transformation, the definition of a degree is required. Due to the calculation of the number of input features in 2.12, only a degree of 2 is used. The 36 columns of the training set result in 703 features for a degree of 2 and 9139 features for a degree of 3. Processing 9139 features requires much more processing power which is not possible in this project. As mentioned earlier also a regression algorithm is chosen to perform polynomial regression. Because ridge is capable of performing an automated feature selection, it is selected to be trained with the polynomial features. To select the most important features, the model would have to apply regularization. This is indicated by a high hyperparameter value tuning the amount to be applied. To verify the performance, also a linear model without regularization is trained.

Contrary to the preceding models, the scores on the test set are slightly worse than on the training set, which is the expected behavior. The metrics of the linear implementations trained on the polynomial features show higher performance than the models that were not trained on this. For the test set, an MSE of 0.1222 and an MAE of 0.1067 were calculated as well as an R^2 score of 0.8793. The learning curve shows a training score that gradually decreases to eventually end up close to the cross validation score. The slightly translucent color indicates a fluctuation during the training of the different folds. The solid line only shows the mean of the R^2 scores. This can be seen for the cross validation score, which shows high fluctuations significantly improving towards the end. A reason for the fluctuations could be the number of features that are created. With the increased feature space, it is less likely for a feature to have an impact on the target value. Additionally, noise is added to the features. Because linear regression has no option to apply regularization, automatically selecting the most important attributes, the fluctuations increase.

Ridge regression shows some improvement in the metrics with an MSE of 0.1147 and an R^2 score of 0.8868. The train metrics, on the other hand, are worse than the ones achieved by the linear model, indicating regularization that was applied. This observation is further confirmed by an α value of 10. Due to the high value of α,

it can be confirmed that a large amount of l_2 regularization was introduced to the training. The learning curve also shows much more fluctuations in the training score than the linear model. In the cross evaluation score though, less fluctuation can be identified. A reason for this could be the model applying the regularization to decrease the impact of the less important attributes. It has to be noted that the α value of 10 is the highest regularization the model could choose from. Higher values might have a positive impact on the results by applying even more regularization.

Support Vector Machine

During the training of the SVR, problems arose because the algorithm got stuck and did not finish in time. With the low number of samples and only ten iterations trained, an execution time below 24h is expected. After the algorithm did not finish in this time, the parameter grid was adapted only using the default kernel. This lead to the algorithm converging in a reasonable amount of time.

The model trained this way performs better than the linear models using polynomial features with an R^2 score of 0.9311, an MSE of 00736 and an MAE of 0.0381. Out of the reduced parameter space, the best parameter combination found has a *tol* value of 0.0001, which in regards to the default of 1e-3 is slightly lower. On the other hand, the *epsilon* value (ϵ) of 0.01 is lower than the default of 0.1.[37] The parameter defines the distance between the two support vectors in which as much data points as possible should be fit. The low value indicates that either a lot of outliers are expected or the data points are all clustered in a small area. By default, C is set to 1, tuning the amount of penalty used in the error function. The evaluated model has a value of 5 which is significantly higher, showing that regularization or penalty leads to high performance for the SVM. This penalty can also be seen in the learning curve, which shows a training error that in comparison to the other models is further away from 1.

Random Forest

The random forest regressor achieves an MSE of 0.0054, an MAE of 0.0347 and an R^2 of 0.9948 on the training set. These results can be considered very good with the R^2 score very close to 1. Inspecting the parameters found by the randomized search, the random forest trained can be outlined. The *n_estimators* parameter was set to 10, representing "[the] number of trees in the forest [...]" and being equal scikit-learn's default.[38] The trees are further specified by the *max_depth*, *min_samples_leaf* and *min_sample_split* determining the maximum depth a tree can grow and the number of minimum samples a leaf needs to exist as well as the minimum number of samples a leaf has to hold before being able to be split.[cf. 38] The criterion used to evaluate

the split is represented by the *criterion* parameter which is set to use the MSE.[cf. 38] The *bootstrap* parameter indicates that bootstrapping is used in the training process, meaning that to train the trees, different samples are used to better represent the information embedded in the data.[cf. 38] If no bootstrapping is used, the trees would all represent the most significant features of the dataset, because they would produce the lowest error, completely ignoring the information encapsulated in the rest of the data. The learning curve of the random forest can be considered quite normal with the training score starting higher and getting closer to the cross validation with the number of samples used for training.

By implementing the gradient boosting variant of the random forest algorithm, the XGBoost model achieves a better result. The results on the training set are an MSE of 0.0038, an MAE of 0.0288 and an R^2 score of 0.9948. Comparing the hyperparameters found by the randomized search to the ones found for the random forest, it is discovered that the number of estimators trained is 50 times higher with a value of 500. This means that more trees were being trained, being able to model even more patterns of the data. This could be the reason for the increase in the R^2 score and the decrease in error. Another difference is the *max_depth* which with a value of 2 is lower than the value of 15 chosen for the random forest. A small value usually is considered better as overfitting is decreased due to the lower complexity of the model. It can be argued though that because only 10 estimators were trained for the random forest, the value has to be higher in order to fit the data sufficiently. The other parameters are not provided by random forest implementation of scikit-learn and therefore have to be evaluated independently. First is the *learning_rate* parameter, representing the learning rate of boosting. With its value of 0.5, it is higher than the default value of 0.3.[cf. 39] This means that the model can learn patterns faster by increasing the value the weights get adapted at each iteration. For the *booster* parameter, the default value *gbtree* is chosen, indicating that a tree-based boosting is used.[cf. 39] The boosting is further improved by applying regularization. The amount of regularization used is determined by the *reg_alpha* and *reg_lambda* parameters, representing the l_1 and l_2 norm respectively. With the chosen values of 0.7 and 0.1, it can be said that a large amount of l_1 and a very small amount of l_2 regularization is used to train the model. The learning curve of the XGBoost is similar to random forest but shows significantly more fluctuation in the cross validation error across the whole curve. Due to its low amplitude of roughly 0.01, the fluctuation is ignored.

Artificial Neural Networks

The ANN scores an MSE of 0.0346 and an MAE of 0.422 with an R^2 score of 0.9600 on the test set. These results are better than the linear models but worse than the random forest models. The scale of the learning curve for this model is different because the

R^2 score had to be modified using only the minuend of the equation. This way, the algorithm only has to reduce the function to 0 to achieve an R^2 score of 1. Analyzing the curve, high fluctuations on both the training and cross validation score can be identified. During the training process, fluctuations were discovered that lead to a non-reproducibility of the model. This means that every time the model is (re)trained, a different score is achieved. Even setting the random state for several parts of the algorithm did not change this. [Brownlee 40] discusses some of the reproducibility issues with Keras on his website. He also offers several ideas to explore but does not offer a definite solution.[cf. 40] Because the scope of this thesis is limited, no solution could be found. Therefore, the score can only be seen as a single representation of the algorithm's capability. To describe the ANN trained in more detail, the parameter combination found by the randomized search has to be analyzed. For the optimizer, the default value *adam* was chosen. The *batch_size*, indicating the number of samples used for a single training step, was set to 16 and *epoch*, the number of epochs that should be trained, to 100. The number of hidden neurons chosen is 8 and therefore fits the optimal number calculated by 3.5.

To investigate the impact of the number of neurons chosen, two separate models are trained using the number of neurons calculated by both equations. In this case, no variation of grid search is used to not influence the results. [Heaton 17]'s approach has to be modified by dropping the third rule. This rule would limit the number of hidden neurons to four because the output layer only consists of two neurons. To see if a large number of neurons would impact the model in a negative way, an additional model is trained with 100 hidden layers. Analyzing the R^2 score of the different setups, it can be seen that 8 neurons lead to the best results on both datasets. The other two implementations lead to very similar results. It can be said that in this case, 8 neurons is expected to be the optimal amount of neurons to choose. An increase in the number of neurons though does not impact the model as much as expected. A more detailed configuration and scores can be found in A.24.

Because a model with a single hidden layer already reaches an R^2 score close to 1, training a DNN is not considered. By adding another layer, the model would be capable of fitting the data even better, which in this case would lead to overfitting. As discussed earlier, this is generally not wanted.

Ranking

Before training, it was decided to use the R^2 score as the metric of choice to evaluate the models. To be more specific, the score achieved on the test set as it is the closest representation to reality and therefore the generalization capability of the model. After the scores are collected, the models are ranked to see which ones performed best. The ranking shows that the random forest models performed best. It can also be confirmed

that the XGBoost implementation outperforms random forest, even if it's just by a very small amount. The two models are followed by the ANN. Even though there were problems with its reproducibility, it is still ranked third. Because it reaches a score that is 0.08 higher than its successor, even small fluctuations won't affect its rank. Shortly after, the SVR can be found followed by the less complex models. The linear models using polynomial features to inflate the feature space perform fairly well. They are also better than their corresponding models without polynomial features.

To sum up the ranking, it can be said that the more complexity an algorithm is able to model, the higher the model performance achieved. Under the context of overfitting, the high complexity is still to be preferred, as long as train and test score are close to each other. The test score is still higher than the training score of the models with less complexity. Therefore it can be said that in this project, a model that is capable of modeling high complexity is to be preferred because it can reach a high generalization capability.

The ranking also allows drawing a line between the models that fulfill the requirements defined in the business understanding and those that do not. By doing so, a clear separation between the linear models and the more complex ones can be observed. This further supports the preceding statement that the models of high complexity should be preferred.

Correlation Matrix

Because against the assumption that the data is not sufficient, very high scores are achieved. This leads to an assumption, that a single value might correlate high with the targets. To investigate this, a correlation matrix is given in A.57. The matrix shows the correlation of the target attributes with the features used for training. Analyzing the matrix, the correlation of *zu verdampfendes Wasser pro* m^2 to the first target stands out. With a value of 0.96, the correlation can be considered very high. Both the attribute and the target value are based on calculations made by ResMa. Investigating the calculation of the target, it is discovered that the high correlating attribute is used to calculate the target. This indicates a positive impact of the prior feature selection by the business. To further support this assumption, the calculation of the attribute has to be investigated. If the values used to calculate the attribute are already present in the feature space used for training, they can be dropped as their information is already incorporated. Indeed, *Bandgeschwindigkeit ABB1 [BSV]* and *Gipsverbrauch aktuell kg/min* are used in the calculation. It can be stated that for future projects, those attributes can be dropped if *zu verdampfendes Wasser pro* m^2 is used as it represents an engineered feature already including the information.

Another observation that can be made is that most attributes show a very low correlation with the targets. Only 5 attributes correlate stronger than 0.1 with the first target value. For the second target value, 24 attributes can be identified this way. This means that the number of features used to predict the first attribute can be reduced by a lot. Because of the imbalance in correlations, it can also to be considered to predict both attributes independently from each other to reduce the feature space as much as possible for both targets.

Besides those finding it can be seen that the engineered features that were produced by splitting the product names show the highest correlation with the targets. The engineering steps are considered successful and should be performed in the future projects.

4.5 Evaluation

4.5.1 Project Review

As suggested in the concept, the process so far has to be reviewed in order to process the findings and to prepare them for a presentation and discussion with the business. Before being able to conduct the business understanding process, an actual state analysis was conducted, describing the current situation and forming a hypothesis of what to achieve by performing a CRISP-DM. Out of the findings gathered, the business objectives were developed in close cooperation with the business partners. The object in the center of the analysis was identified as the plasterboard dryer, which showed great potential for optimization. Also, machine learning practices were chosen as the tool to conduct the analysis and build a model for deeper insights. It was discovered that the business shows great interest in the process as a whole and how improvements for future implementations can be made. Furthermore, the energy consumption is selected as the target of choice and insights should be gathered regarding the values and relationships representing the target. To complete the phase, a system architecture is discussed to perform the analysis and process the data. Due to a high demand in computational power and security concerns, a decision for an on-premise solution instead of the cloud was made.

After setting up the environment, the data was exported from the system and statistical as well as visualization techniques were used to produce a data description and exploration report supplemented by an EDA. In those, valuable findings were discovered that were of value for the business and for the following steps. It also pointed out the issue of missing values being overly present in the gathered data and an insufficient amount of data regarding the target attributes.

Using this information about the data, an attempt was made to improve data quality and to prepare the data for the modeling process. Supporting this process, packages were used to parallelize the process as much as possible to decrease the runtime of the algorithms. To further optimize the processing time, it was decided to order the data preparation steps by the amount of potential data dropped in descending and amount of potential processing power required in ascending order. In the first step, columns that were identified as unnecessary or that did not provide a sufficient amount of data were dropped from the dataset. Then values were propagated, dealing with the missing values of most columns by taking over the values of preceding filled in values. Incorporating the domain knowledge provided by the business, certain values were then cropped to proactively remove outliers. Columns that were identified to be of categorical type were then encoded and one attribute containing a string of information was split into several categorical attributes. To prepare the data for the merging process, the timestamps were updated according to the domain knowledge about the process of value recording. Furthermore, they were aggregated so that each row represented a quarter-hour. In an attempt to further improve the data quality, outlier detection was performed on the columns including potential outliers. The data were merged and missing values created by this process were filled using an imputation strategy. Finally, the data were scaled using standardization.

In the next phase, the prepared data was used to train several models that were selected by making assumptions about the task to be performed, such as identifying the problem as a multi-output regression. Each model was trained using a randomized search and a 5 fold cross validation. For each model type, a parameter grid was created, providing possible values for performance influencing parameters. The best parameter combination was then used to train a model that was evaluated on the basis of the MSE, MAE, and R^2. Furthermore, a learning curve for the cross validation was plotted to show how R^2 evolves over the course of training. [41]

4.5.2 Discussion

After reviewing the project, the results have to be presented to and discussed with the business. Going over the statistical and visual analysis together with the responsible business partners, it became clear that the domain knowledge provided in the business understanding phase was incorporated sufficiently. By confronting the business with the insights gathered, the issue of insufficient data quality was concretized. Several visualizations lead to an identification of attributes as not representative of reality. Reasons for this were discussed, reaching from erroneous calculations and malfunctioning sensors up to insights that were not communicated prior to the project. Those issues were documented and are being discussed in the following also providing solutions for most of them. Nonetheless part of the project was to identify those problems

and pitfalls. Also, the insights gathered are of great value for the business. Because a detailed description of those insights is not in the focus of this thesis but rather the process, they are not discussed further but categorized as *valuable*.

The problems occurred can be divided into different categories representing different objects in the process. The first object of analysis is the ResMa, more specific the data export. Due to the slow Internet connection and the limited export capability of the system, the export process turned out to be more difficult than expected. Being able to only export a small number of attributes at once and being tied to certain time spans depending on the compression strength chosen, the possible export scenarios are restricted in a way that makes future exports of large amounts of data even more difficult. This issue could be resolved by either improving the Internet connection of the system or physically exporting a dump of the underlying SQL database. The latter seems like the best approach as the export becomes more comfortable. A downside could be that no live data can be processed this way without deploying the model directly on the plant's server. Another pitfall is the deletion of old records. By setting, the system deletes data that is older than roughly two years. If in the future analysis are planned on a larger time frame, this setting should be adapted as otherwise no sufficient information about the past can be gained. Last, the calculations made by the system as well as the sensors data should be verified manually to guarantee a high quality of the data output. If calculations already include inaccuracies, the machine learning model has no chance of predicting the output correctly.

The second object of discussion is the data itself. Before forwarding the data to the data analyst, the business should go over the attributes, documenting the domain knowledge like the expected range and the plausibility of certain values. Because the data analyst usually lacks domain knowledge and heavily relies on the business input, this is considered very important as it can drastically improve the data understanding and preparation steps by detecting outliers more effectively or to identify malfunctioning sensors earlier in the process. Looking at the data, many missing values and a high number of outliers were detected. These together with the target variables not represented sufficiently, heavily impact the whole process in a negative way. Most of those problems could have been avoided by the business performing those steps mentioned. Using the timestamp as an index seemed like a good approach at the beginning of the process, but turned out to be difficult as the system exported the data with different compression strengths. A solution was developed in corporation with the business to adjust the system and its sensors to record values every minute or second. This way, merging the data becomes easier, because it is more likely for each timestamp to be represented in each file exported. In addition, the analysis of the correlation matrix hints that the feature space can be reduced even further. Because also both targets show a high correlation with different attributes, a suggestion can be made to perform a separate regression for each target instead of a multi-output.

The last object to be mentioned is the process structure of the company. Even though a standard process for determining the data confidentiality exist, the business partners are not trained sufficiently to apply those to the required data. Also, cloud computing is still a new territory for which no guidelines and standard processes exist. Both issues can lead to time delays in the project because cloud solutions can neither be used nor taken into account.

After conducting the meeting, it becomes clear that the CRISP-DM indeed has to be performed in an iterative way. The proposals made by the business in the evaluation process most certainly affect every phase of the cycle. By conducting multiple iterations of CRISP-DM, those newly gathered insights can be incorporated into the process until a state of concurrence is achieved.

4.5.3 Conclusion

To conclude the evaluation, it has to be decided whether or not the insights gathered are sufficient enough to proceed into the deployment phase. In the preceding paragraphs, the results were presented. It can be seen that the business gained a lot of valuable insights from the process. Those insights include new influencing factors and correlations among attributes which were not expected, but also malfunctioning sensors, erroneous calculation and underperforming export functionality of the ResMa. The high number of problems encountered shows that there is still room for improvement for the entire process. The data understanding can be significantly improved by the new domain knowledge gained from the evaluation phase. Even though the models fulfilled the requirements of the business understanding, they are considered not ready for deployment. The reason for this is that only a small part of the problem was modeled. Also, data from only a single month was used, leading to insufficient representation of reality. Considering these factors, the models should at least be trained on an additional month before a deployment can deliver sufficient value in an optimization process regarding the dryer.

Due to those results, the deployment phase can not be entered yet. It is necessary to return back to the business understanding phase to perform another iteration of the process. This has to be done until a sufficient state is achieved in the evaluation phase. It is important to mention that after performing the evaluation, a close connection to the business understanding phase is discovered. The repeated discussion with the business results in similar outcomes like the development of new understandings and demands out of the findings gathered so far.

5 Evaluation and Discussion

5.1 Cross-Industry Standard Process for Data Mining

Evaluating the conducted CRISP-DM as a whole, it can be said that the process itself is straight forward but highly flexible. By incorporating the business partners and the domain knowledge provided by such, it helps to not lose track of the business objectives. Furthermore, the data understanding and preparation steps, including feature engineering and selection, benefit greatly from those insights. Due to the cyclic structure, a feedback loop is included, which makes the process particularly suitable for companies that conduct their first data mining project. The evaluation phase helps to identify pitfalls and practices that should or should not be kept for future projects.

One thing to notice though is the order of the data understanding and data preparation phase. The process assumes that the data quality of the exported data is sufficient enough to perform a statistical as well as visual analysis on it. If the data is of insufficient quality or is not joined to a single dataset yet, the analysis can still lead to insights that are of value for the data preparation. When it comes to relationships and information the business is interested in, drawbacks have to be accepted. These are caused by missing values, outliers and placeholders which can distort the view. Also, domain knowledge is not yet incorporated into the data. To solve this issue, additional data analysis should be conducted after the data preparation phase. This way, the clean data can provide more valuable insights regarding the business objectives. Additionally, a final observation of the data can help to select the appropriate algorithms and hyperparameters in advance.

5.2 Downsides of Small Datasets

Due to the lack of data and transformations made on it, the dataset used for training in this project is considered fairly small. This can lead to several issues regarding the output models, some of which are discussed in the following.

By using a low number of training samples, the algorithm is more prone to overfitting the data than with a high number of samples. The reason for this is that less complexity has to be modeled, meaning that the same number of parameters can fit the data more precisely. By doing so, a higher training score is achieved but the generalization capability of the model on new data, e.g. the training set, decreases significantly. In a

deployed state, the model will most likely encounter data it has not seen before, making overfitting in the training process of high impact on the production performance.

Another issue is the lack of representativeness of reality. In this project e.g., only a single month is used as a representation of reality. This can result in a model that represents the given data very well, but makes many mistakes on new data. As mentioned before, a deployed model usually only encounters unseen data, which becomes a problem as predictions are made based on the patterns found in the single month used for training. To avoid this, every month should be represented equally and of multiple years to cover as many patterns as possible.

5.3 Conclusion and Outlook

Even though the deployment phase was not reached in this project, it is still considered successful. In view of the fact that the use case covered in this thesis is considered as the first data mining project in the company, crucial pitfalls were identified that can help improve future projects. To now evaluate the project in regards to the formulated hypothesis, it can be said that by performing CRISP-DM and utilizing machine learning, indeed valuable insights about the influencing factors of the energy consumption of the plasterboard dryer could be gained. It was also done by using the data provided by the ResMa. Therefore, the hypothesis can be accepted. During the business understanding phase of the CRISP-DM, objectives were defined that go beyond the hypothesis. The achievement of those has extended the scope of possibilities for future business objectives by presenting state of the art techniques to the business partners.

The next step would be to conduct the next cycle of the CRISP-DM and to keep iterating until the deployment phase can be entered. For the upcoming iteration, the focus should lie on collecting more data especially regarding the time frame recorded. By doing so, more representative models can be created. Problems that were documented during this thesis should also be taken into account in order to improve the process. To reduce the workload, implementations used in this project can be reused and slight adaptations should be made to the parts regarding the data understanding phase. If different findings are produced there, the data preparation step has to be modified as well. As long as the data mining goal does not change, the modeling process can be kept due to its implementation featuring randomized search for the parameter selection and a dynamic number of in- and output features.

The future approach to optimize the energy consumption of the dryer is to cooperate with *SAS* analytics and software solutions [42] to develop analytic and predictive models for every plant of the Knauf group. Those models should then be rolled out

worldwide to optimize the plants at scale. To support this cooperation, the findings of this thesis can be used to avoid pitfalls already identified. Reusing the analysis of the data should also be considered to get a first overview of the relationships between the attributes. The trained models though should only be reused after verifying the performance on new data. Since CRISP-DM has proven to be a good fit for the company and this type of problem, it is suggested for the projects ahead.

Bibliography

[1] K. G. KG, *Knauf - Our History*, 2019. [Online]. Available: https://www.knauf.com/en/about-knauf/our-history/ (visited on 06/19/2019).

[2] H. Wintrich, "Energieeffizienz , ein wichtiger Schritt zur Standortsicherung bei Knauf Energiemanagement bei Knauf", [Online]. Available: https://www.mainfranken.org/media/www.mainfranken.org/org/med%7B%5C_%7D50199/52040%7B%5C_%7Dpraesentation%7B%5C_%7Dwintrich.pdf.

[3] I. Steinwart and A. Christmann, *Support Vector Machines*. 2008. DOI: 10.1007/978-0-387-77242-4.

[4] R. Wirth and J. Hipp, "CRISP-DM : Towards a Standard Process Model for Data Mining", *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, 2000. DOI: 10.1.1.198.5133.

[5] F. Schafer, C. Zeiselmair, J. Becker, and H. Otten, "Synthesizing CRISP-DM and Quality Management: A Data Mining Approach for Production Processes", *2018 IEEE International Conference on Technology Management, Operations and Decisions, ICTMOD 2018*, pp. 190–195, 2019. DOI: 10.1109/ITMC.2018.8691266.

[6] A. Géron, *Hands On Machine Learning with Scikit-Learn and TensorFlow*, First Edit. 2017. DOI: 10.3389/fninf.2014.00014. arXiv: 1412.3919.

[7] P. Harrington, *Machine Learning in Action*. 2012, ISBN: 9781617290183.

[8] M. T. Mitchell, *Machine Learning*. 1997, ISBN: 0070428077. [Online]. Available: http://www-stat.stanford.edu/%7B~%7Dtibs/book/preface.ps.

[9] S. Marsland, *Machine Learning: An Algorithmic Perspective*. 2014, ISBN: 9781466583337.

[10] M. F. Henrik Brink, Joseph Richards, *Real-World Machine Learning*. 2016, ISBN: 9781617291920.

[11] D. H. Wolpert, "The Lack of A Priori Distinctions Between Learning Algorithms", 1996. DOI: 10.1162/neco.1996.8.7.1341.

[12] J. Patterson and A. Gibson, *Deep learning: A Practionar Approach*. 2017. DOI: 10.1038/nature14539. arXiv: arXiv:1312.6184v5. [Online]. Available: http://www.nature.com/doifinder/10.1038/nature14539.

[13] D. Scikit-learn, *SVM: Maximum margin separating hyperplane*, 2019. [Online]. Available: https://scikit-learn.org/stable/auto%7B%5C_%7Dexamples/svm/plot%7B%5C_%7Dseparating%7B%5C_%7Dhyperplane.html (visited on 08/08/2019).

[14] J. Hearty, J. Huffman, and A. Pajankar, *Advanced Machine Learning with Python*. 2016, p. 278, ISBN: 978-1-78439-863-7. [Online]. Available: www.packtpub.com.

[15] X. Developers, *XGBoost Documentation*, 2016. [Online]. Available: https://xgboost.readthedocs.io (visited on 05/24/2019).

[16] B. Nikhil and N. Locascio, *Fundamentals of Deep Learning book*. O'Reilly, 2017, ISBN: 9781491925614.

[17] J. Heaton, *Introduction to Neural Networks with Java, 2nd Edition*. 2008, p. 440. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[18] GTI, *ResMa*, 2018. [Online]. Available: https://gti.de/produkte-der-gti/resma/ (visited on 06/17/2019).

[19] H. Yip, P. J. F. Groenen, and G. Nalbantov, "SmartEDA: An R Package for Automated Exploratory Data Analysis", *JSS Journal of Statistical Software*, vol. VV, no. 10, 2018. DOI: 10.18637/jss.v000.i00. arXiv: arXiv:1607.04543v1. [Online]. Available: https://rdrr.io/cran/SVMMaj/f/inst/doc/paper.pdf.

[20] B. Badie, D. Berg-Schlosser, and L. Morlino, *International Encyclopedia of Political Science*. SAGE Publications, 2011, ISBN: 9781483305394. [Online]. Available: https://books.google.de/books?id=Vn2iCQAAQBAJ.

[21] R. D. Peng and E. Matsui, "The Art of Data Science: A guide for anyone who works with Data", *Journal of Chemical Information and Modeling*, vol. 53, p. 160, 2016. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[22] S. Jaggi, "Descriptive Statistics and Exploratory Data Analysis", *Indian Agricultural Statistics Research Institute*, 2013. [Online]. Available: http://iasri.res.in/design/ebook/EB%7B%5C_%7DSMAR/e-book%7B%5C_%7Dpdf%20files/Manual%20II/1-Descriptive%20Statistics.pdf.

[23] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature Selection: A Data Perspective", vol. 50, no. 6, 2016. DOI: 10.1145/3136625. arXiv: 1601.07996. [Online]. Available: http://arxiv.org/abs/1601.07996%7B%5C%%7D0Ahttp://dx.doi.org/10.1145/3136625.

[24] D. Scikit-learn, *Documentation*, 2019. [Online]. Available: https://scikit-learn.org/stable/documentation.html (visited on 07/30/2019).

[25] ——, *Regression metrics*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/model%7B%5C_%7Devaluation.html%7B%5C#%7Dregression-metrics (visited on 05/20/2019).

[26] T. Masters, *Practical Neural Network Recipies in C++*. 2016. DOI: 10.1016/c2009-0-22399-3. [Online]. Available: https://books.google.de/books?redir%7B%5C_%7Desc=y%7B%5C&%7Dhl=de%7B%5C&%7Did=7Ez%7B%5C_%7DPq0sp2EC%7B%5C&%7Dq=number+of+hidden+nods%7B%5C#%7Dv=onepage%7B%5C&%7Dq=pyramid%20rule%7B%5C&%7Df=false.

[27] P. S. Foundation, *Multiprocessing*, 2019. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html (visited on 07/17/2019).

[28] Celeryproject.org, *Billiard*, 2019. [Online]. Available: https://github.com/celery/billiard (visited on 07/17/2019).

[29] M. Waskom, *Seaborn*, 2018. [Online]. Available: https://seaborn.pydata.org/ (visited on 07/19/2019).

[30] D. Scikit-learn, *Overview of outlier detection methods*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/outlier%7B%5C_%7Ddetection.html%7B%5C#%7Doverview-of-outlier-detection-methods (visited on 07/23/2019).

[31] ——, *Comparing anomaly detection algorithms for outlier detection on toy datasets*, 2019. [Online]. Available: https://scikit-learn.org/stable/auto%7B%5C_%7Dexamples/plot%7B%5C_%7Danomaly%7B%5C_%7Dcomparison.html%7B%5C#%7Dcomparing-anomaly-detection-algorithms-for-outlier-detection-on-toy-datasets (visited on 07/23/2019).

[32] ——, *SimpleImputer*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html (visited on 07/23/2019).

[33] ——, *Imputation of missing values*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/impute.html%7B%5C#%7Dimputation-of-missing-values (visited on 07/23/2019).

[34] ——, *IterativeImputer*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html (visited on 07/23/2019).

[35] ——, *Pipeline*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html (visited on 07/25/2019).

[36] ——, *LinearRegression*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear%7B%5C_%7Dmodel.LinearRegression.html (visited on 07/30/2019).

[37] ——, *SVR*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html (visited on 08/12/2019).

[38] ——, *RandomForestRegressor*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html%7B%5C#%7Dsklearn.ensemble.RandomForestRegressor (visited on 07/31/2019).

[39] X. Developers, *XGBoost Parameters*, 2019. [Online]. Available: https://xgboost.readthedocs.io/en/latest/parameter.html (visited on 07/31/2019).

[40] J. Brownlee, *How to Get Reproducible Results with Keras*, 2019. [Online]. Available: https://machinelearningmastery.com/reproducible-results-neural-networks-keras/ (visited on 07/08/2019).

[41] D. Scikit-learn, *Plotting Learning Curves*, 2019. [Online]. Available: https://scikit-learn.org/stable/auto%7B%5C_%7Dexamples/model%7B%5C_%7Dselection/plot%7B%5C_%7Dlearning%7B%5C_%7Dcurve.html (visited on 07/30/2019).

[42] S. I. Inc., *SAS - The Power To Know*, 2019. [Online]. Available: https://www.sas.com/en%7B%5C_%7Dus/home.html (visited on 06/08/2019).

[43] Anaconda, *Anaconda*, 2019. [Online]. Available: https://www.anaconda.com/ (visited on 06/14/2019).

[44] Project Jupyter, *Jupyter*, 2019. [Online]. Available: https://jupyter.org (visited on 06/14/2019).

[45] N. Developers, *NumPy.org*, 2019. [Online]. Available: https://www.numpy.org/ (visited on 06/18/2019).

[46] S. Developers, *SciPy.org*, 2019. [Online]. Available: https://www.scipy.org/ (visited on 06/18/2019).

[47] The Matplotlib development team, *Matplotlib.org*, 2018. [Online]. Available: https://matplotlib.org/ (visited on 06/18/2019).

[48] P. Developers, *Pandas.pydata.org*, 2019. [Online]. Available: https://pandas.pydata.org/ (visited on 07/19/2019).

[49] K. Developers, *Keras.io*, 2019. [Online]. Available: https://keras.io/ (visited on 06/18/2019).

[50] ——, *Wrappers for the Scikit-Learn API*, 2019. [Online]. Available: https://keras.io/scikit-learn-api/ (visited on 06/18/2019).

[51] TensorFlow, *TensorFlow Github*, 2019. [Online]. Available: https://github.com/tensorflow/tensorflow (visited on 06/18/2019).

[52] ——, *TensorFlow.org*, 2019. [Online]. Available: https://www.tensorflow.org/ (visited on 06/18/2019).

[53] X. Developers, *XGBoost Scikit-Learn API*, 2019. [Online]. Available: https://xgboost.readthedocs.io/en/latest/python/python%7B%5C_%7Dapi.html%7B%5C#%7Dmodule-xgboost.sklearn (visited on 07/31/2019).

A Appendix

A.1 Business Understanding

A.1.1 Time Benchmark

Data was generated with scikit-learn's *make_classification* function and shape (800000,10). The parameters *n_informative* and *n_redundant* were set to 2 and 0 respectively. When required, a *random_state* of 0 was used. *Accuracy* rounded to three decimal places. *Training Time* rounded to two decimal places. *Training Time* is relative to the machine trained on but gives an approximate understanding of the time complexity of each model.

For the DNNs, *batch_size* of 5 and 500 *epochs* are chosen. The activation function for the hidden layers is *ReLu* and *sigmoid* for the output layer. As a loss function *binary cross entropy* and as optimizer *adam* is used. The layer configuration can be seen below. Both networks have 10 inputs and 1 output with hidden layers in between. Number of neurons per layer must be read from left (input) to right (output).

Table A.1: Detailed Time Benchmarks

Model	**Configuration**	**Accuracy**	**Training Time**
SGD Classifier	default	0.912	0.92
Random Forest Classifier I	n_estimators=100; max_depth=10	0.924	21.05
Random Forest Classifier II	n_estimators=1000; max_depth=10	0.924	146.28
XGBoost Classifier I	n_estimators=100; max_depth=10	0.928	94.52
XGBoost Classifier II	n_estimators=1000; max_depth=10	0.927	851.99
Deep Neural Network I	neurons=10,8,6,1	0.924	101981.91
Deep Neural Network II	neurons=10,8,1	0.927	110117.01

A.1.2 Resource Requirements Report

Introduction to the problem and justification for the resources needed

Training a machine learning model consist of a preparation and a training step.

The preparation step includes exploring the data and getting insight using visualization techniques. This step does not require much processing power because all analysis are performed in a static way.

On the other hand, the training step does indeed require a lot of resources. In this step an algorithm is using the training data to train a model. This means, that the more data is used for the process, the longer it takes. Additionally the algorithm has to be executed several times so the model can represent all connections between the data effectively. Training a neural network takes such a long time that most of the times a GPU is required to finish training in a reasonable amount of time. This is due to the calculations that involve huge matrices and vector that have to be multiplied, transposed or inversed. Designed for matrix operations, a GPU can speed up the process.

Both steps require a significant amount of memory, because the data has to be made available to the training algorithm. In this case memory is required over storage, because the access time significantly speeds up the process.

To sum it up, the preparation step only needs a lot of memory, but can be performed without significant processing power. The training step does not only require a lot of memory, but also needs a lot of processing power in the form of CPUs or GPUs to make the algorithm finish in a reasonable amount of time (minutes to hours instead of hours and days).

In the following the resources needed, the options available to guarantee those and a cost estimation for the solutions is made. Depending on the solution, data protection compliances and the location the data is stored at are analyzed.

Resources needed for each Task of the Data Mining Process:

Task	Processor	RAM	GPU/TPU support
Data Exploration and Preparation	Medium	High	No
Modeling (simple)	High	High	No / Yes (XGBoost)
DNN	Very High	High	Yes (Nvidia CUDA, CC > 3.0)

Viable **Options** for each Task:

Task	On Premise	Cloud Company	Cloud Location	Cloud Instance Name	Cloud Instanc Pricing $/h
Data Exploration and Preparation					
	web01 dev server	-	-	-	-
	-	AWS	Germany	c5.4xlarge (EC2)	0.88
	-	Azure	Germany	E8 v3	0.69
Modeling (simple)					
	-	AWS	Germany		
				p3.2xlarge (EC2)	3.8
				ml.p2.2xlarge (SageMaker)	1.85
		Azure	North Europe		
				NC6	0.97
				NV6	2.03
DNN					
	-	AWS	Germany	cf. Modeling	cf. Modeling
	-	Azure	North Europe	cf. Modeling	cf. Modeling

Notes:
The modeling task can also be performed on a weaker machine. However, it is debatable if setting up a new environment for the DNN tasks is worth the small savings in costs.

When evaluating price, functionality and location, the AWS ml.p2.2xlarge, which is part of the SageMaker program (Machine Learning), seems like a great fit. The instance is therefore used for further cost estimations.

Compliance **Data Protection** Germany AWS
https://aws.amazon.com/de/compliance/germany-data-protection/

Compliance **Data Protection** Azure
https://azure.microsoft.com/en-us/global-infrastructure/germany/
https://www.microsoft.com/en-us/trust-center/privacy/data-location

Notes:
AWS offers its viable instances in Germany where it also stores the data. Azure on the other hand only provides North Europe as a location for its instances, making it unclear in which country the data is stored. Both solutions comply with the European laws for storing data.

Estimated Time needed for each Task:

Task	**Time needed in % with no respect to the instance type**
Data Exploration and Preparation	70%
Modeling (simple)	20%
DNN	10%

Cost Estimation Using Preferred Machines (6 weeks, 8h/day):

Task	**Machine**	**Cost Estimation in $**
Data Exploration and Preparation	web01 dev server	0.70 * 240 * 0.1 = **16.80**
Modeling (simple)	ml.p2.2xlarge	0.20 * 240 * 1.85 = **88.80**
DNN	ml.p2.2xlarge	0.1 * 240 * 1.85 = **44.40**
Total		16.80 + 88.80 + 44.40 = **150**

Notes:
Because the time is an estimation and small costs for data transfer and storage might arise, a budget of **250$** is necessary to perform the tasks efficiently.

Additional Notes:
If the data is considered classified, the machine learning process can include various steps to anonymize the data to prevent any conclusions about the input data. For example the data can be scaled, resulting in a mean of 0 and a standard deviation of 1 (Standard Scaling) or scaling the data into a range of -1 and 1 (MinMax Scaling). Also the attribute names and categorical attributes can be replaced by integers. This way the data could be processed in the cloud without any concerns about a 3rd party successfully recovering any conclusions about the input data from it. On the other hand, a table held in the local network, matching the integers with the replaced strings, could interpret the result.

Example of **Data Preparation**:

Source: https://dev.to/r0f1/a-simple-way-to-anonymize-data-with-python-and-pandas-79g

Initial Dataset:

Out[3]: (891, 12)

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Output Dataset **Anonymization**:

Out[24]:

	0	1	2	3	4	5	6	7
0	0	1	0	2	1	2	16.846655	11.866699
1	1	3	1	1	0	2	22.967661	6.523808
2	1	1	0	0	1	2	34.873726	2.272420
3	0	2	0	0	0	2	9.149005	2.408852
4	0	3	0	0	0	2	32.161475	6.477172

Output Dataset **Scaling**:

Out[17]:

	0	1	2	3	4	5	6
0	-1.730108	-0.789272	0.827377	-0.530377	0.432793	-0.473674	-0.502445
1	-1.726220	1.266990	-1.566107	0.571831	0.432793	-0.473674	0.786845
2	-1.722332	1.266990	0.827377	-0.254825	-0.474545	-0.473674	-0.488854
3	-1.718444	1.266990	-1.566107	0.365167	0.432793	-0.473674	0.420730
4	-1.714556	-0.789272	0.827377	0.365167	-0.474545	-0.473674	-0.486337

Notes:
The examples above show, that if the preprocessing is done correctly, it is impossible for a 3rd party to draw conclusion about the initial dataset out of the output dataset. The preparation steps do not require a lot of processing power and thus can be performed on a local machine or server.

A.2 Data Understanding

A.2.1 Environment Setup

As operating system *Ubuntu 18.04.2 LTS* should be used. To perform actions on the data and to train the models, the *anaconda* environment version 2019.03, an "Enterprise Data Science Platform" with *Python* version 3.6 is selected due to its already existing familiarity.[43] It also offers an installer specifically for Power8 and Power9 systems.[43] The environment supports *Jupyter* and enables the editing and executing of *Jupyter notebooks*. "The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text."[44] In these notebooks, code snippets can be executed without the need of the whole code to be run. In combination with the markdown support, this enables the user to debug and document the code more comfortable.

A.2.2 Attributes per File

```
abnahme
        ['Schmierung Tisch 1',
        'Summe Stromverbrauch Abnahme',
        'Trockenausschuss [BSA]',
        'Restfeuchte Platten [BSA]',
        'Anzahl Pakete Abnahme',
        'Doppler 2 Bänder',
        'Laufzeit Abnahme']

artikelnummer
        ['Knauf Stadtoldendorf.02 Bandstraße.512-Vorbereitungsstation ...
        ... .Artikelnummer / Baugruppe [BSV]',
        'Ende']

trockner
        ['Gesamtleistung Brenner [BST]',
        'Heizwert Erdgas Hu',
        'Laufzeit Trockner [BST]',
        'Nassausschuss [BST]',
        'Summe Gasverbrauch Trockner [BST]',
        'Summe Stromverbrauch Trockner [BST]',
        'Trocknerleistung']

trockner_effizienz
        ['spez. therm. Energie pro m^2) [BST]',
        'therm. Trockner Effizienz [BST]',
        'zu verdampfendes Wasser pro m^2 [BST]',
        'Gipsverbrauch aktuell kg/min',
        'Plattenbreite in m [BSV]',
        'Reinheitsgrad Gips']

trocknergetriebe
        ['Öltemperatur',
        'Viskosität',
        'Dielektrizitäts-Konstante',
        'Leitfähigkeit']
```

Table A.2: Attributes per File Part I

```
vorbereitungsstation
        ['Laufzeit Mischer [BSV]',
        'Laufmeter Bandstraße',
        'Summe Stromverbrauch Vorbereitungssation',
        'Plattenbreite [BSV]',
        'Bandgeschwindigkeit ABB1 [BSV]',
        'Artikelnummer / Baugruppe [BSV]',
        'Bandgeschwindigkeit ABB2 [BSV]',
        'Bandgeschwindigkeit ABB1 Soll [BSV]']

waerme_rueckgewinnung
        ['Klappenstellung Abluft Zone 02 [BST]',
        'Klappenstellung Abluft Zone 01 zu WRG SOLL [BST]',
        'Klappenstellung Abluft Zone 01 zu WRG [BST]',
        'Klappenstellung Abluft Zone 02 SOLL [BST]',
        'Unterdruck Zone 1 Tür 2 [BST]',
        'Klappenstellung Abluft Zone 02 zu WRG SOLL [BST]',
        'Klappenstellung Abluft Zone 02 zu WRG [BST]',
        'Differenzdruck WT Frischluft [BST]',
        'Ablufttemperatur vor WRG [BST]',
        'Ablufttemperatur nach WRG [BST]',
        'Frischlufttemperatur nach WRG [BST]',
        'Klappenstellung Abluft Zone 01 SOLL [BST]',
        'Klappenstellung Abluft Zone 01 [BST]',
        'Frischlufttemperatur vor WRG [BST]']

zone01
        ['Regel Temperatur Zone 01 SOLL [BST]',
        'Regel Temperatur Zone 01 [BST]',
        'Temperatur Zone 01 Einblasseite [BST]',
        'Temperatur Zone 01 Ausblasseite [BST]',
        'Mittlere Temperatur Zone 01 [BST]',
        'Feuchtebeladung Luft Zone 01 [BST]',
        'Brennerleistung Zone 01 [BST]',
        'Frequenz Umluftventilator re. Zone 01 [BST]',
        'Frequenz Umluftventilator li. Zone 01 [BST]']
```

Table A.3: Attributes per File Part II

```
zone02
        ['Regel Temperatur Zone 02 [BST]',
        'Plattenfeuchte Zone 02 [BST]',
        'Regel Temperatur Zone 02 SOLL [BST]',
        'Temperatur Zone 02 Einblasseite [BST]',
        'Temperatur Zone 02 Ausblasseite [BST]',
        'Feuchtebeladung Luft Zone 02 [BST]',
        'Brennerleistung Zone 02 [BST]',
        'Frequenz Umluftventilator li. Zone 02 [BST]',
        'Mittlere Temperatur Zone 02 [BST]',
        'Frequenz Umluftventilator re. Zone 02 [BST]']

zone03
        ['Regel Temperatur Zone 03 [BST]',
        'Plattenfeuchte Zone 03 [BST]',
        'Regel Temperatur Zone 03 SOLL [BST]',
        'Temperatur Zone 03 Einblasseite [BST]',
        'Temperatur Zone 03 Ausblasseite [BST]',
        'Feuchtebeladung Luft Zone 03 [BST]',
        'Brennerleistung Zone 03 [BST]',
        'Frequenz Umluftventilator li. Zone 03 [BST]',
        'Mittlere Temperatur Zone 03 [BST]',
        'Frequenz Umluftventilator re. Zone 03 [BST]']

zone04
        ['Regel Temperatur Zone 04 SOLL [BST]',
        'Regel Temperatur Zone 04 [BST]',
        'Temperatur Zone 04 Einblasseite [BST]',
        'Temperatur Zone 04 Ausblasseite [BST]',
        'Plattenfeuchte Zone 04 [BST]',
        'Frequenz Umluftventilator li. Zone 04 [BST]',
        'Brennerleistung Zone 04 [BST]',
        'Mittlere Temperatur Zone 04 [BST]',
        'Frequenz Umluftventilator re. Zone 04 [BST]']
```

Table A.4: Attributes per File Part III

A.2.3 Data Description and Exploration Report

```
count    1802.000000
mean        0.268172
std         0.438620
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         1.000000
Name: Schmierung Tisch 1, dtype: float64
```

Table A.5: Statistical Analysis: *Schmierung Tisch 1*

```
Time
2018-11-03 02:11:00    0.0
2018-02-17 10:28:00    0.0
2019-04-17 08:08:00    0.0
2018-08-23 09:59:00    1.0
2017-07-12 10:00:00    1.0
2018-03-11 10:00:00    1.0
2018-04-09 10:00:00    1.0
2018-03-29 10:01:00    1.0
2018-04-04 07:16:00    0.0
2018-11-03 22:36:00    0.0
Name: Schmierung Tisch 1, dtype: float64
```

Table A.6: Sample: *Schmierung Tisch 1* (sample size 10)

```
count    58830.000000
mean         0.106867
std          0.122651
min          0.000000
25%          0.000000
50%          0.078000
75%          0.174667
max          2.000000
Name: Restfeuchte Platten [BSA], dtype: float64
```

Table A.7: Statistical Analysis: *Restfeuchte Platten [BSA]*

```
count    52866.000000
mean        31.190976
std         28.374517
min          0.000000
25%          0.000000
50%         32.000000
75%         44.000000
max        364.000000
Name: Anzahl Pakete Abnahme, dtype: float64
```

Table A.8: Statistical Analysis: *Anzahl Pakete Abnahme*

```
Time
2018-02-02 18:45:00    24.0
2019-03-27 17:15:00    80.0
2018-01-11 02:15:00     0.0
2019-05-26 15:00:00     0.0
2017-11-29 11:15:00     0.0
2019-09-06 05:45:00     0.0
2018-11-13 10:15:00    36.0
2018-12-22 21:45:00     0.0
2018-12-01 09:45:00    36.0
2019-08-03 01:30:00    52.0
Name: Anzahl Pakete Abnahme, dtype: float64
```

Table A.9: Sample: *Anzahl Pakete Abnahme* (sample size 10)

```
count    542424.000000
mean          0.499440
std           0.066189
min           0.000000
25%           0.500000
50%           0.500000
75%           0.500000
max           1.000000
Name: Doppler 2 Bänder, dtype: float64
```

Table A.10: Statistical Analysis: *Doppler 2 Bänder*

```
Time
2017-07-11 00:04:00    0.500000
2018-06-11 14:08:00    0.500000
2018-03-05 03:41:00    0.500000
2019-01-28 21:32:00    0.500000
2018-07-31 13:50:00    0.500000
2017-10-28 13:20:00    0.428571
2018-11-23 23:36:00    0.500000
2019-06-18 02:37:00    0.500000
2018-06-15 19:25:00    0.500000
2017-09-12 12:50:00    0.500000
Name: Doppler 2 Bänder, dtype: float64
```

Table A.11: Sample: *Doppler 2 Bänder* (sample size 10)

```
DatetimeIndex(['2017-01-11 00:00:00', '2017-01-11 00:15:00',
'2017-01-11 00:30:00', '2017-01-11 00:45:00',
'2017-01-11 01:00:00', '2017-01-11 01:15:00',
'2017-01-11 01:30:00', '2017-01-11 01:45:00',
'2017-01-11 02:00:00', '2017-01-11 02:15:00',
...
'2019-12-06 23:50:00', '2019-12-06 23:51:00',
'2019-12-06 23:52:00', '2019-12-06 23:53:00',
'2019-12-06 23:54:00', '2019-12-06 23:55:00',
'2019-12-06 23:56:00', '2019-12-06 23:57:00',
'2019-12-06 23:58:00', '2019-12-06 23:59:00'],
dtype='datetime64[ns]', name='Time', length=565632, freq=None)
```

Table A.12: Datetime Index

```
count    98142.000000
mean      7654.124261
std       6298.486188
min          0.000000
25%          0.000000
50%      10750.000000
75%      13641.133789
max      16970.533203
Name: Gesamtleistung Brenner [BST], dtype: float64
```

Table A.13: Statistical Analysis: *Gesamtleistung Brenner [BST]*

```
count    63308.000000
mean     11865.657788
std       3395.000366
min         14.200000
25%      11214.949951
50%      13465.000000
75%      13837.000000
max      16970.533203
Name: Gesamtleistung Brenner [BST], dtype: float64
```

Table A.14: Statistical Analysis: *Gesamtleistung Brenner [BST]* without zeros

```
count    98142.000000
mean         2.380451
std          3.200707
min          0.000000
25%          0.000000
50%          2.730000
75%          2.862667
max         60.192665
Name: Trocknerleistung, dtype: float64
```

Table A.15: Statistical Analysis: *Trocknerleistung*

```
count    40136.000000
mean         3.241472
std          1.514151
min          0.031235
25%          2.716808
50%          2.811971
75%          3.150590
max         66.698669
Name: spez. therm. Energie pro m2 [BST], dtype: float64
```

Table A.16: Statistical Analysis: *spez. therm. Energie pro* m^2 *[BST]* without zeros

	Öltemperatur	Viskosität	Dielektrizitäts-Konstante	Leitfähigkeit
count	50187.000000	50189.000000	752492.000000	50186.000000
mean	35.006116	143.237751	2.400266	6.886346
std	13.405775	106.783255	0.978874	0.500006
min	0.000000	0.000000	0.000000	0.000000
25%	23.000000	56.200001	1.410000	6.800000
50%	40.000000	142.266663	2.640000	6.800000
75%	45.000000	190.600006	3.260000	6.820000
max	62.000000	894.333313	3.990000	31.673334

Figure A.1: Statistical Analysis: *trocknergetriebe*

	Laufzeit Mischer [BSV]	Plattenbreite [BSV]	Bandgeschwindigkeit ABB1 [BSV]	Artikelnummer / Baugruppe [BSV]	Bandgeschwindigkeit ABB2 [BSV]	Bandgeschwindigkeit ABB1 Soll [BSV]
count	58497.000000	423.000000	98139.000000	467.000000	58493.000000	71501.000000
mean	59.755642	873.785602	35.228099	308.920771	33.895414	58.862695
std	7.181056	307.958003	29.610086	179.619185	29.529528	12.950845
min	0.000000	600.000000	0.000000	71.000000	0.000000	6.000000
25%	59.947338	600.000000	0.000000	152.000000	0.000000	59.850000
50%	60.000000	625.000000	48.400002	181.000000	40.020000	65.250001
75%	60.103161	1250.000000	65.333336	451.000000	65.800003	65.900002
max	568.944763	1250.000000	79.699997	888.000000	80.900002	82.699997

Figure A.2: Statistical Analysis: *vorbereitungsstation*

	Klappenstellung Abluft Zone 02 [BST]	Klappenstellung Abluft Zone 01 zu WRG SOLL [BST]	Klappenstellung Abluft Zone 01 zu WRG [BST]	Klappenstellung Abluft Zone 02 SOLL [BST]	Unterdruck Zone 1 Tür 2 [BST]	Klappenstellung Abluft Zone 02 zu WRG SOLL [BST]	Klappenstellung Abluft Zone 02 zu WRG [BST]
count	74.0	267.000000	1978.000000	254.000000	78215.000000	805.000000	779.000000
mean	0.0	97.283396	54.660668	0.640748	-0.086134	28.396791	31.431911
std	0.0	11.831224	17.322420	6.639587	0.079127	21.048473	16.734910
min	0.0	17.500000	19.000000	0.000000	-0.600278	0.000000	7.333333
25%	0.0	100.000000	54.099998	0.000000	-0.145500	11.000000	19.800000
50%	0.0	100.000000	58.368420	0.000000	-0.051359	23.000000	28.285715
75%	0.0	100.000000	62.272728	0.000000	-0.029000	40.000000	40.500000
max	0.0	100.000000	100.000000	100.000000	0.528323	100.000000	100.000000

Figure A.3: Statistical Analysis: *waerme_rueckgewinnung* part I

Differenzdruck WT Frischluft [BST]	Ablufttemperatur vor WRG [BST]	Ablufttemperatur nach WRG [BST]	Frischlufttemperatur nach WRG [BST]	Klappenstellung Abluft Zone 01 SOLL [BST]	Klappenstellung Abluft Zone 01 [BST]	Frischlufttemperatur vor WRG [BST]
39579.000000	98142.000000	98142.000000	98142.000000	1248.000000	9510.000000	98025.000000
5.206028	119.727914	81.201084	100.543274	17.604474	24.279182	29.662923
2.479054	55.347512	36.023618	47.942178	12.824975	10.131576	9.866311
0.000000	7.866667	4.333333	6.000000	0.000000	0.000000	0.000000
4.250000	60.000000	44.533333	53.000000	11.000000	20.454546	24.000000
4.850000	156.000000	100.000000	128.000000	18.000000	24.555555	32.000000
6.050000	161.000000	111.000000	142.000000	24.000000	28.333334	37.000000
19.488890	204.066666	123.599998	160.000000	100.000000	100.000000	63.799999

Figure A.4: Statistical Analysis: *waerm_rueckgewinnung* part II

```
count    254.000000
mean       0.640748
std        6.639587
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max      100.000000
Name: Klappenstellung Abluft Zone 02 SOLL [BST], dtype: float64
```

Table A.17: Statistical Analysis: *Klappenstellung Abluft Zone 02 SOLL [BST]*

```
count      5.00000
mean      32.55000
std       38.60068
min        2.75000
25%       15.00000
50%       20.00000
75%       25.00000
max      100.00000
Name: Klappenstellung Abluft Zone 02 SOLL [BST], dtype: float64
```

Table A.18: Statistical Analysis: *Klappenstellung Abluft Zone 02 SOLL [BST]* without zeros

	Regel Temperatur Zone 01 SOLL [BST]	Regel Temperatur Zone 01 [BST]	Temperatur Zone 01 Einblasseite [BST]	Temperatur Zone 01 Ausblasseite [BST]	Mittlere Temperatur Zone 01 [BST]	Feuchtebeladung Luft Zone 01 [BST]	Brennerleistung Zone 01 [BST]	Frequenz Umluftventilator re. Zone 01 [BST]	Frequenz Umluftventilator li. Zone 01 [BST]
count	5493.000000	98141.000000	98138.000000	98139.000000	882158.000000	98112.000000	98142.000000	96425.000000	96425.000000
mean	300.965962	201.396219	201.394894	119.326803	132.381224	168.576532	2637.491212	29.502863	29.503219
std	31.119664	123.166687	123.167706	59.065626	68.041221	228.973100	2262.345244	21.887496	21.887834
min	0.000000	7.000000	7.000000	6.000000	6.000000	2.000000	0.000000	0.000000	0.000000
25%	299.000000	51.000000	51.000000	52.000000	52.000000	4.000000	0.000000	0.000000	0.000000
50%	308.000000	270.200012	270.200012	158.066666	175.000000	11.000000	3831.666748	44.346668	44.353333
75%	315.000000	312.000000	312.000000	164.000000	182.000000	450.533325	4790.000000	47.599998	47.599998
max	368.500000	376.000000	376.000000	232.000000	245.000000	750.000000	5961.000000	50.000000	50.000000

Figure A.5: Statistical Analysis: *zone01*

	Regel Temperatur Zone 02 [BST]	Plattenfeuchte Zone 02 [BST]	Regel Temperatur Zone 02 SOLL [BST]	Temperatur Zone 02 Einblasseite [BST]	Temperatur Zone 02 Ausblasseite [BST]	Feuchtebeladung Luft Zone 02 [BST]	Brennerleistung Zone 02 [BST]	Frequenz Umluftventilator li. Zone 02 [BST]	Mittlere Temperatur Zone 02 [BST]	Frequenz Umluftventilator re. Zone 02 [BST]
count	98138.000000	98138.000000	6336.000000	98138.000000	98138.000000	98138.000000	98138.000000	100558.000000	98138.000000	96892.000000
mean	214.449739	39.252090	314.319576	214.453469	112.332468	325.975688	2575.253905	29.256468	129.279666	29.532112
std	128.872304	6.285206	32.477729	128.872644	52.228501	292.096083	2151.773470	22.073364	62.332077	21.973650
min	8.000000	24.500000	0.000000	8.000000	9.000000	1.000000	0.000000	0.000000	8.000000	0.000000
25%	55.000000	32.619999	307.333333	55.000000	55.000000	11.000000	0.000000	0.000000	59.000000	0.000000
50%	293.200012	41.200001	325.000000	293.200012	143.000000	388.000000	3703.800049	44.386669	165.233330	44.500000
75%	330.000000	43.700001	333.000000	330.000000	147.066666	613.000000	4630.000000	47.599998	176.666672	47.599998
max	400.000000	100.000000	374.000000	400.000000	291.533325	750.000000	6198.133301	50.000000	297.933319	50.000000

Figure A.6: Statistical Analysis: *zone02*

	Regel Temperatur Zone 03 [BST]	Plattenfeuchte Zone 03 [BST]	Regel Temperatur Zone 03 SOLL [BST]	Temperatur Zone 03 Einblasseite [BST]	Temperatur Zone 03 Ausblasseite [BST]	Feuchtebeladung Luft Zone 03 [BST]	Brennerleistung Zone 03 [BST]	Frequenz Umluftventilator li. Zone 03 [BST]	Mittlere Temperatur Zone 03 [BST]	Frequenz Umluftventilator re. Zone 03 [BST]
count	98138.000000	98138.000000	8309.000000	98138.000000	98138.000000	98138.000000	98138.000000	96892.000000	98138.000000	96892.000000
mean	139.770048	20.283660	196.294503	139.773062	99.487570	261.810110	1503.403440	29.604011	121.054582	29.611295
std	74.313630	20.784407	19.629448	74.313899	46.437268	231.169534	1246.573147	22.011161	59.602684	22.008591
min	8.933333	0.100000	0.000000	8.866667	9.000000	2.000000	0.000000	0.000000	9.000000	0.000000
25%	48.000000	0.200000	189.000000	48.000000	46.200001	18.000000	0.000000	0.000000	51.000000	0.000000
50%	185.000000	25.026667	195.000000	185.000000	125.000000	252.000000	2235.000000	45.200001	156.000000	45.200001
75%	198.000000	30.420000	205.000000	198.000000	130.800003	470.000000	2608.000000	47.599998	164.000000	47.599998
max	299.000000	100.000000	290.000000	299.000000	222.533340	750.000000	5027.733398	50.000000	247.199997	50.000000

Figure A.7: Statistical Analysis: *zone03*

	Regel Temperatur Zone 04 SOLL [BST]	Regel Temperatur Zone 04 [BST]	Temperatur Zone 04 Einblasseite [BST]	Temperatur Zone 04 Ausblasseite [BST]	Plattenfeuchte Zone 04 [BST]	Frequenz Umluftventilator li. Zone 04 [BST]	Brennerleistung Zone 04 [BST]	Mittlere Temperatur Zone 04 [BST]	Frequenz Umluftventilator re. Zone 04 [BST]
count	7717.000000	98138.000000	98138.000000	98138.000000	98138.000000	96892.000000	98138.000000	98138.000000	96892.000000
mean	179.232282	126.299068	126.301529	96.638158	10.037991	28.251363	937.845401	106.949088	28.252757
std	17.085273	68.042574	68.042501	46.873143	9.445206	20.908227	760.619580	54.044387	20.907754
min	0.000000	9.000000	9.000000	9.000000	0.000000	0.000000	0.000000	9.000000	0.000000
25%	176.000000	40.000000	40.000000	38.000000	0.180000	0.000000	0.000000	38.000000	0.000000
50%	183.000000	172.000000	172.000000	127.000000	11.400000	43.263332	1365.666626	142.600006	43.276667
75%	187.000000	181.000000	181.000000	131.199997	17.446667	45.099998	1650.000000	148.000000	45.099998
max	300.000000	241.266663	241.266663	193.133331	100.000000	47.000000	2111.000000	204.866669	47.000000

Figure A.8: Statistical Analysis: *zone04*

A.2.4 Exploratory Data Analysis

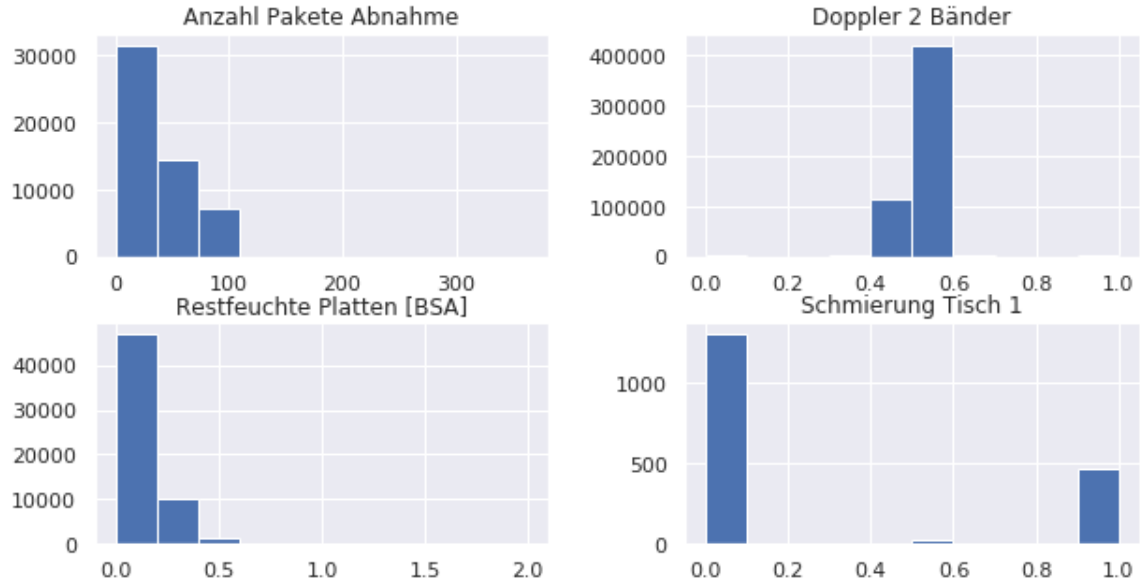


Figure A.9: Histogram: *artikelnummer*

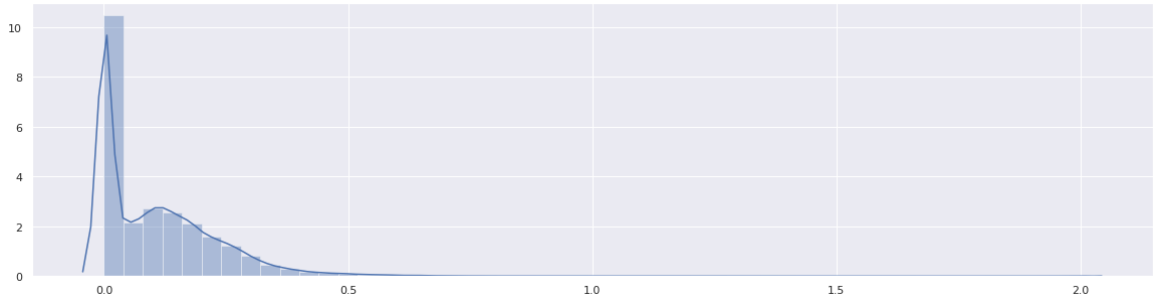


Figure A.10: Distribution Plot: *Restfeuchte Platten [BSA]*

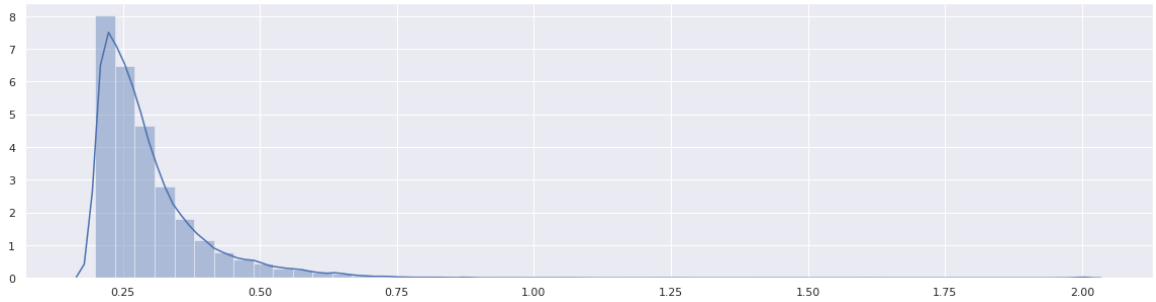


Figure A.11: Distribution Plot: *Restfeuchte Platten [BSA]* cropped > 0.2

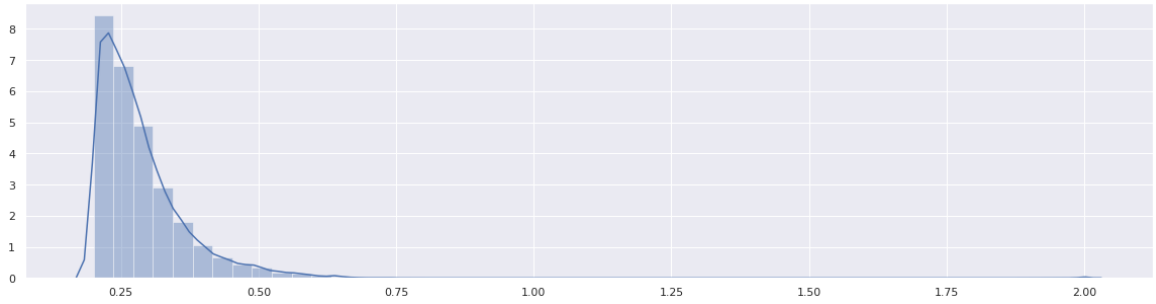


Figure A.12: Distribution Plot: *Restfeuchte Platten [BSA]* cropped, without outliers

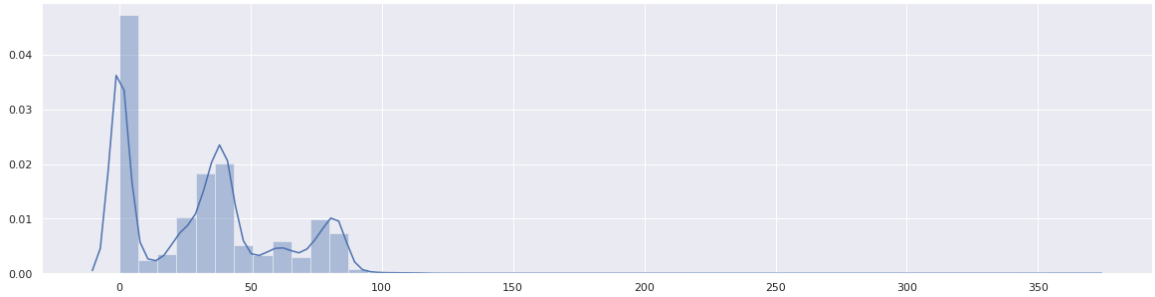


Figure A.13: Distribution Plot: *Anzahl Pakete Abnahme*

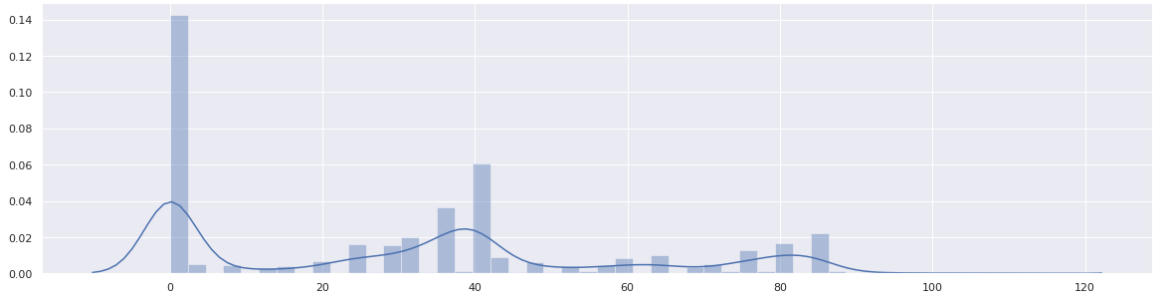


Figure A.14: Distribution Plot: *Anzahl Pakete Abnahme* without outliers

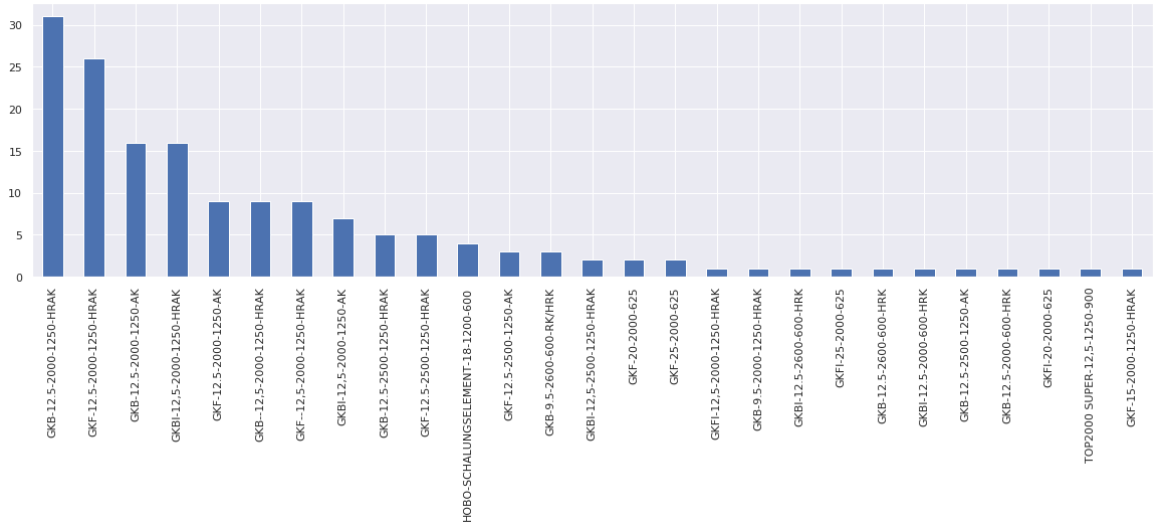


Figure A.15: Distribution Plot: *artikelnummer* count

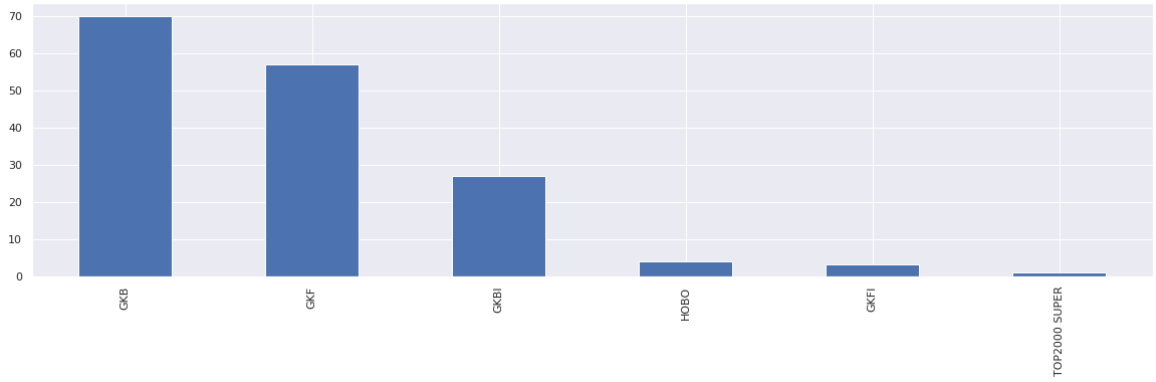


Figure A.16: Distribution Plot: *artikelnummer* transformed product type count

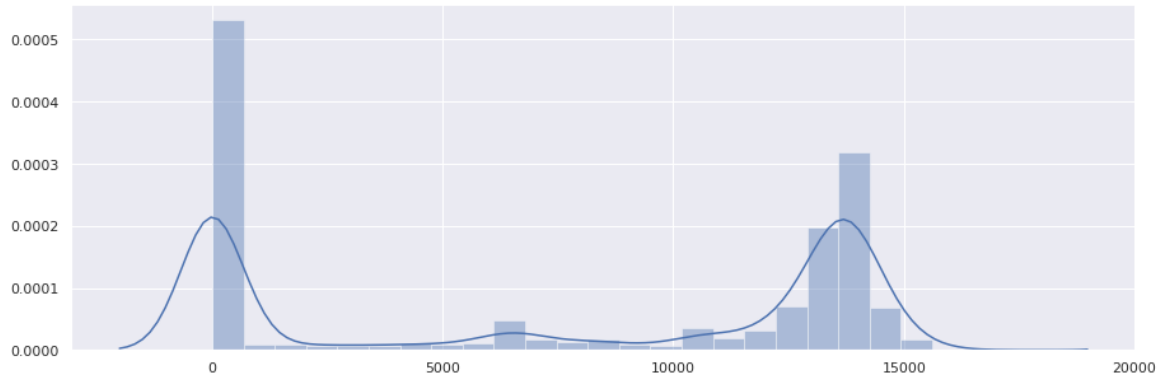


Figure A.17: Distribution Plot: *Gesamtleistung Brenner [BST]*

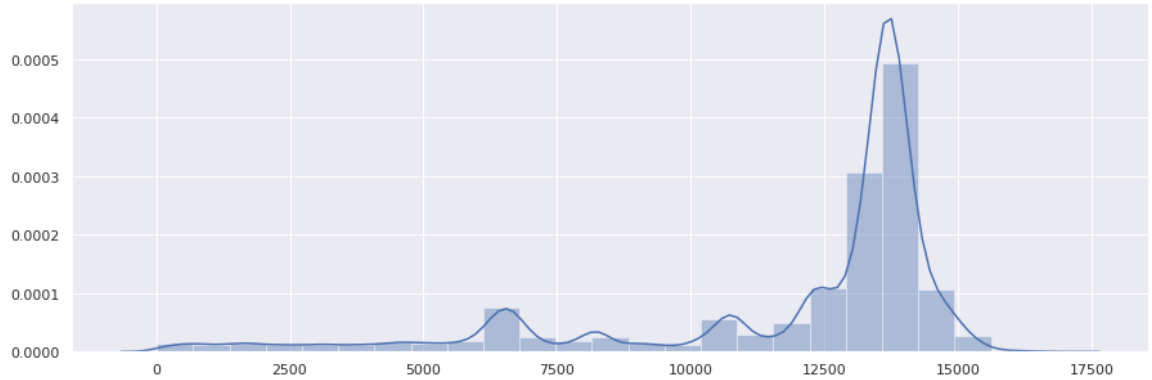


Figure A.18: Distribution Plot: *Gesamtleistung Brenner [BST]* without zeros

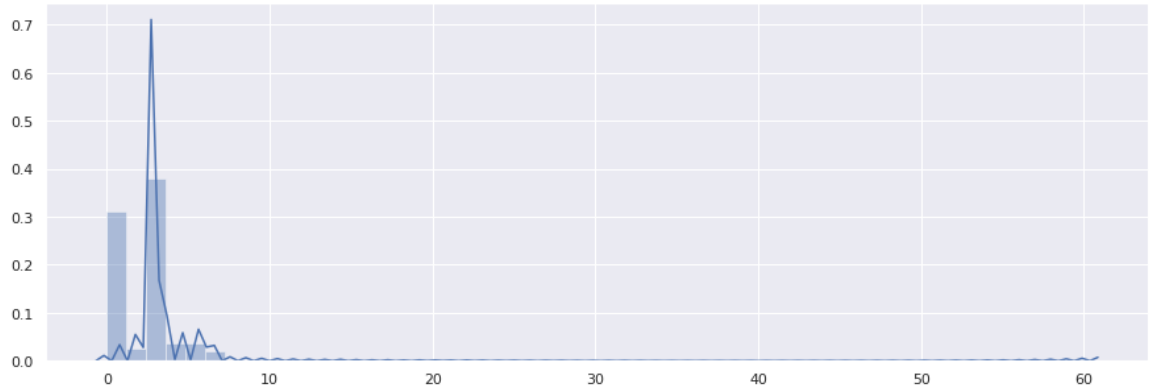


Figure A.19: Distribution Plot: *Trocknerleistung*

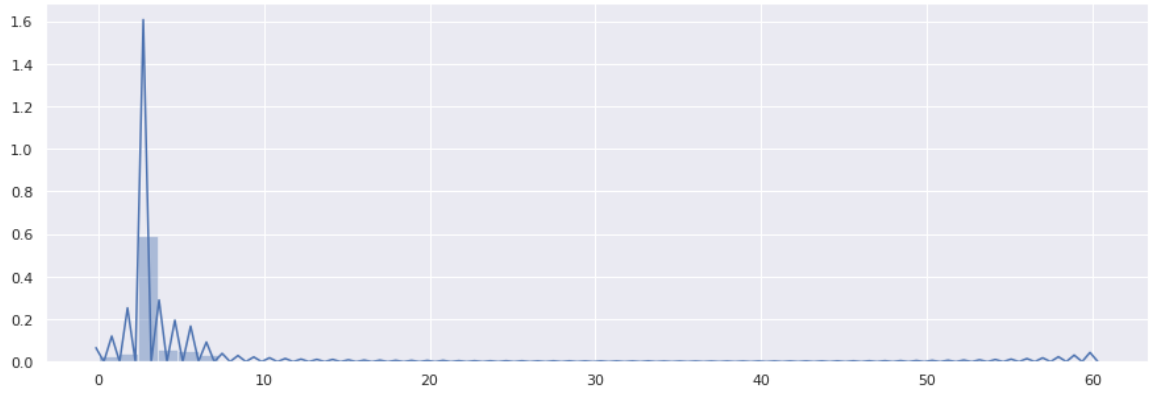


Figure A.20: Distribution Plot: *Trocknerleistung* without zeros

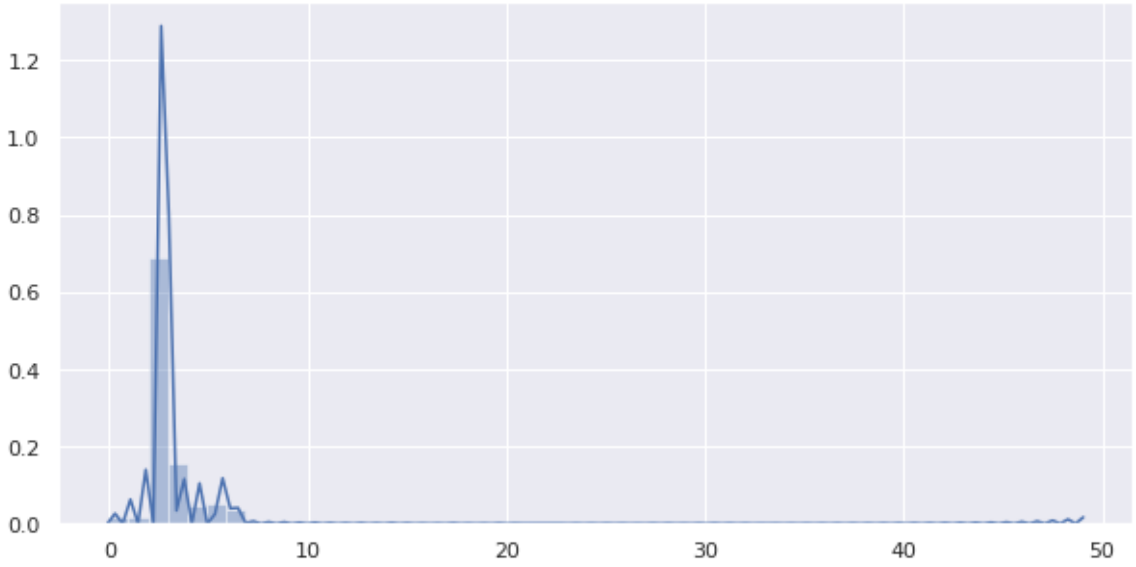


Figure A.21: Distribution Plot: *Trocknerleistung* without zeros and outliers

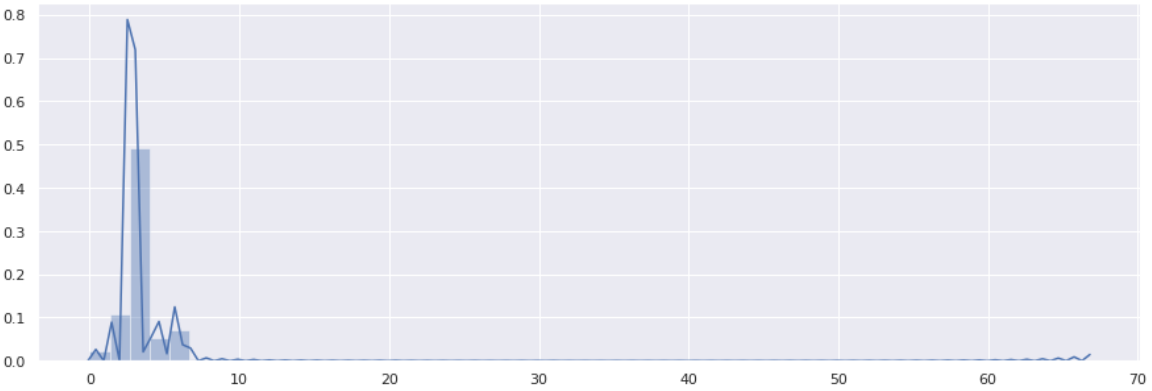


Figure A.22: Distribution Plot: *spez. therm Energie pro* m^2 *[BST]*

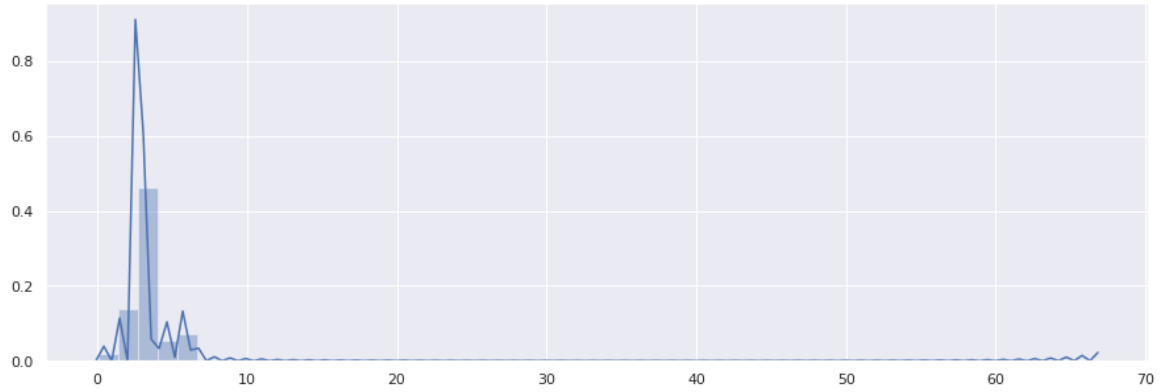


Figure A.23: Distribution Plot: *spez. therm Energie pro* m^2 *[BST]* without zeros

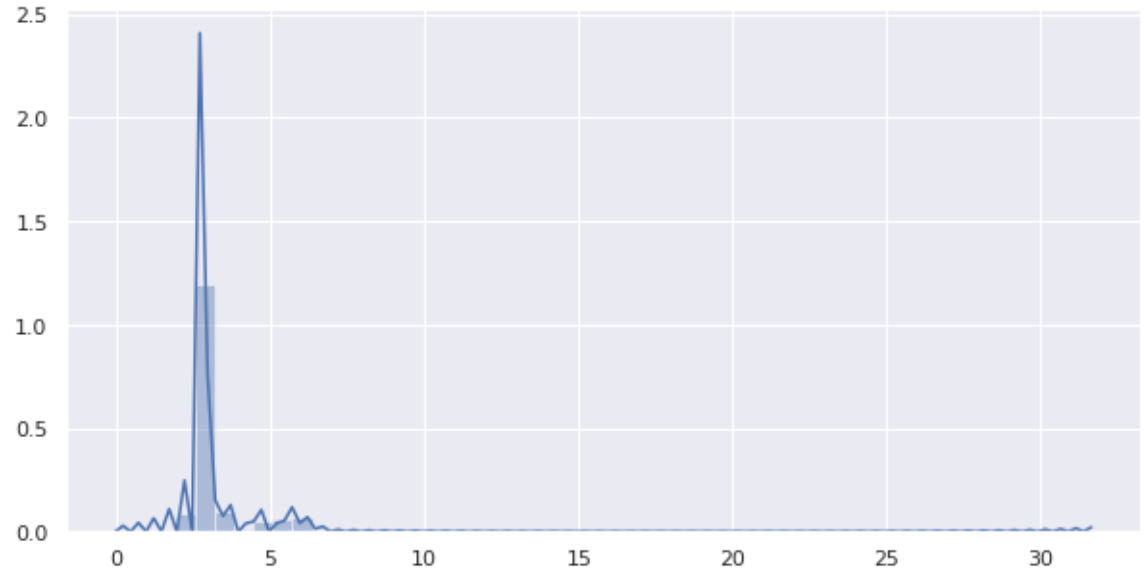


Figure A.24: Distribution Plot: *spez. therm Energie pro* m^2 *[BST]* without zeros and outliers

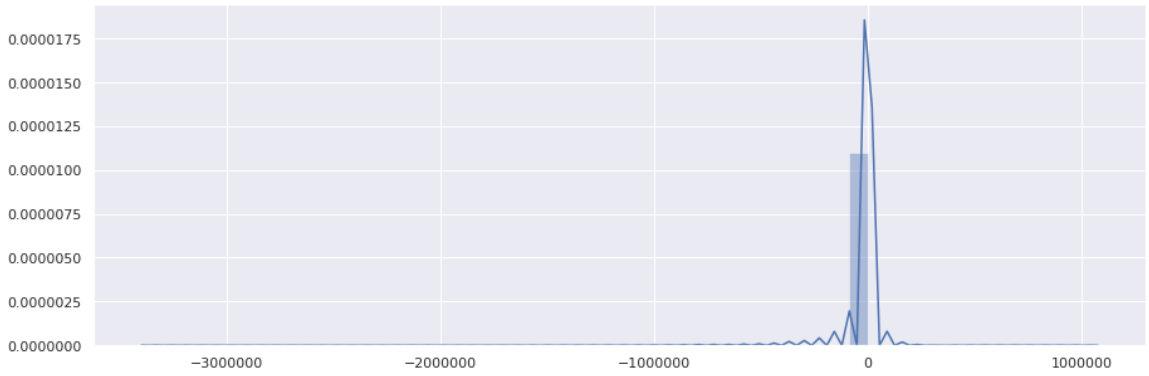


Figure A.25: Distribution Plot: *therm. Trockner Effizienz [BST]*

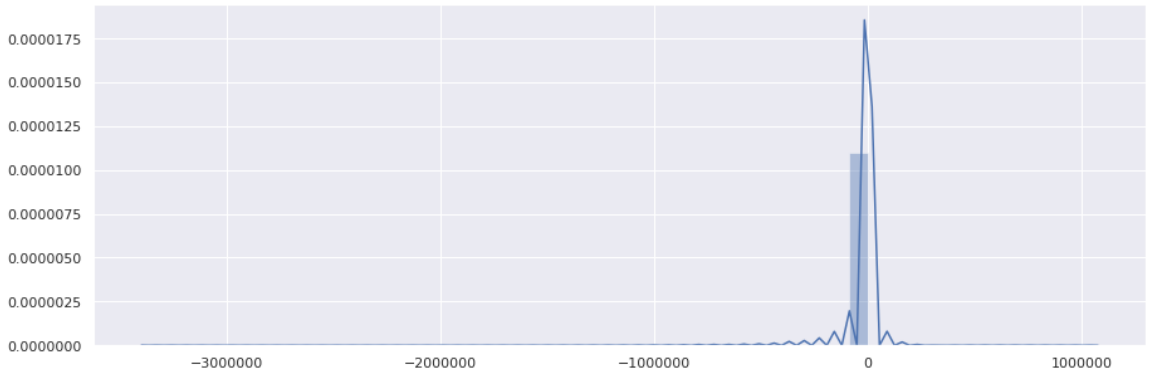


Figure A.26: Distribution Plot: *therm. Trockner Effizienz [BST]* without zeros

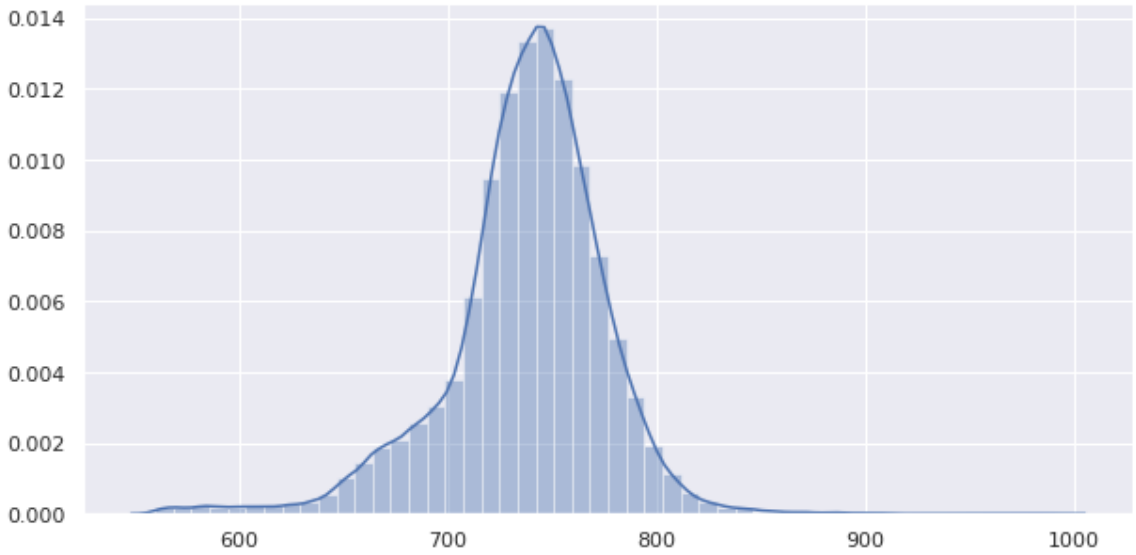


Figure A.27: Distribution Plot: *therm. Trockner Effizienz [BST]* cropped 560 < x < 1000

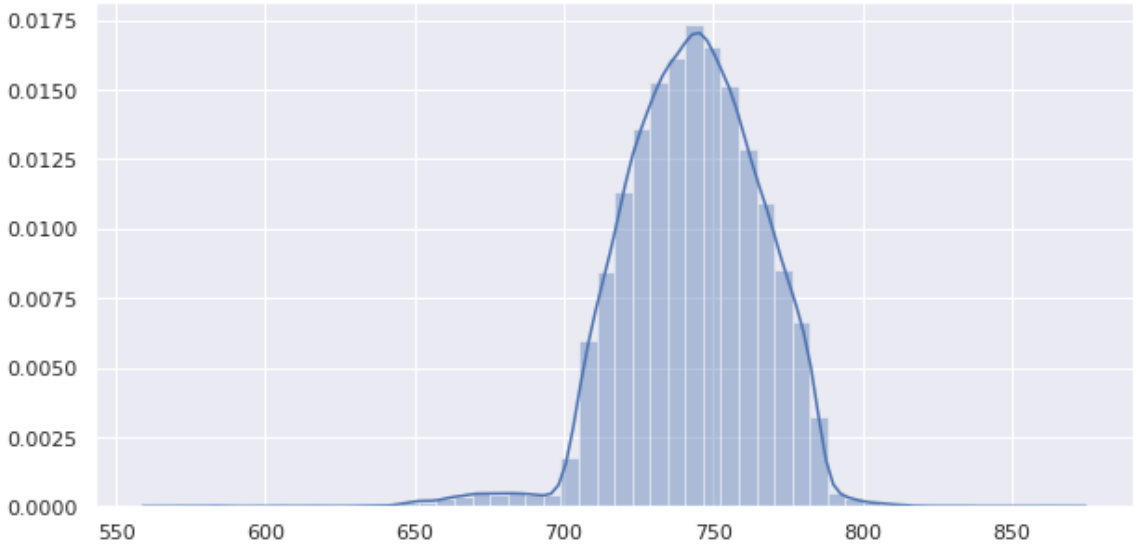


Figure A.28: Distribution Plot: *therm. Trockner Effizienz [BST]* cropped without outliers

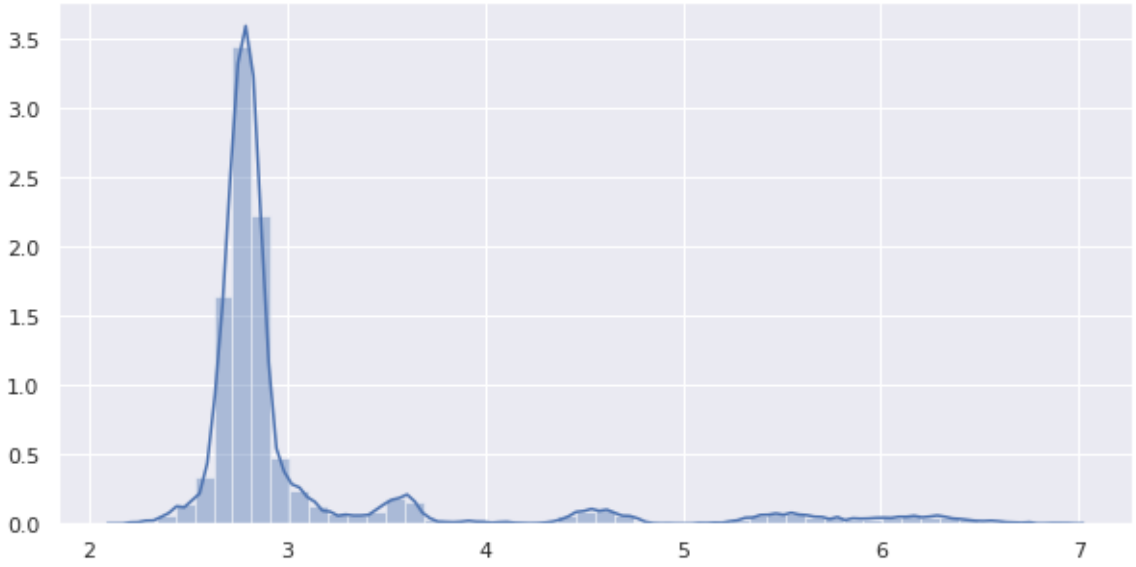


Figure A.29: Distribution Plot: *spez. therm Energie pro* m^2 *[BST]* merged with *therm. Trockner Effizienz [BST]*

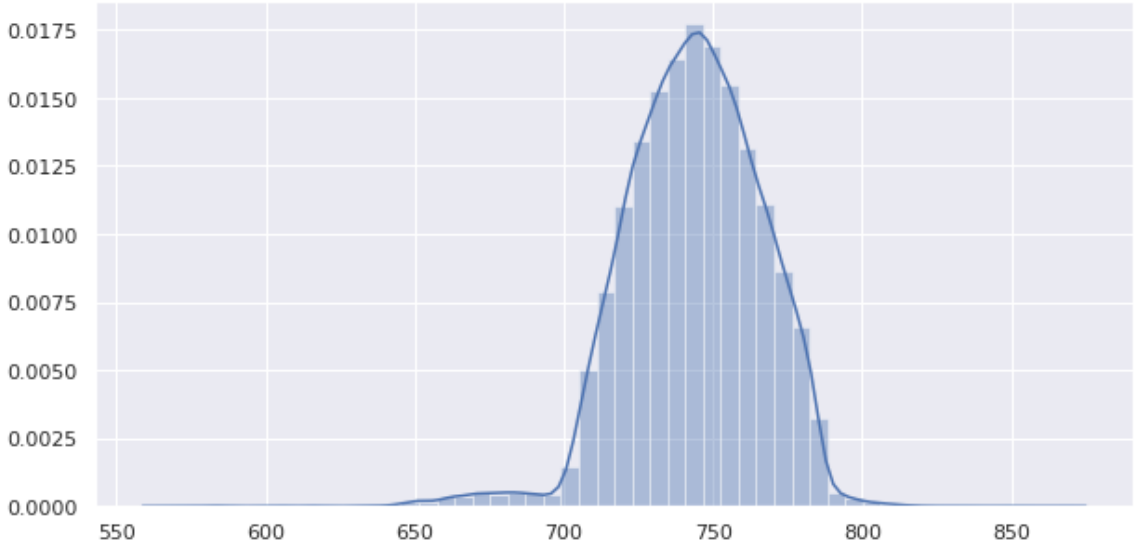


Figure A.30: Distribution Plot: *therm. Trockner Effizienz [BST]* merged with *spez. therm Energie pro* m^2 *[BST]*

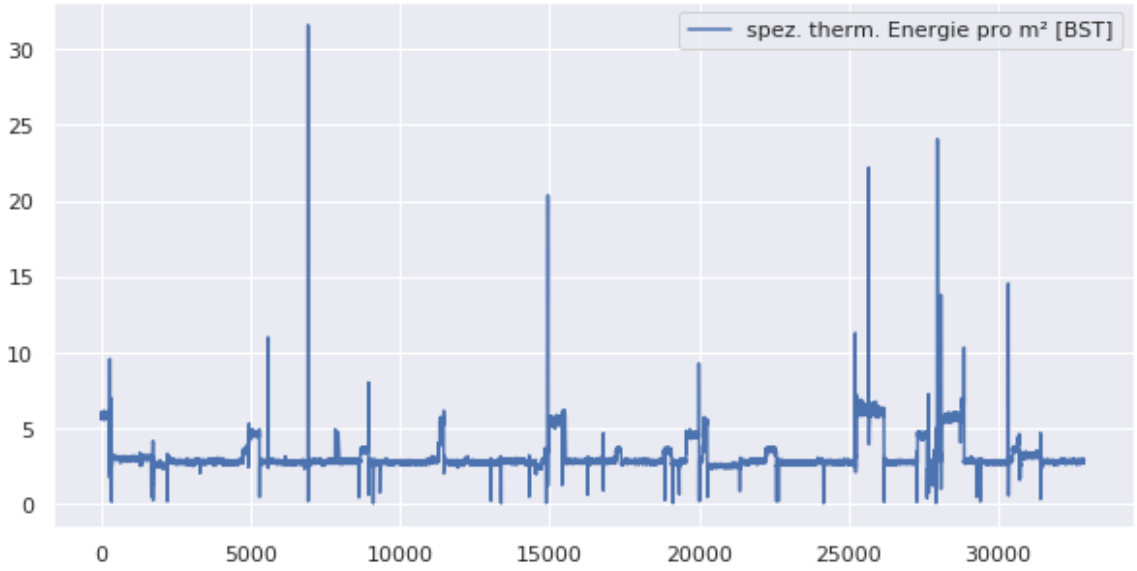


Figure A.31: Line Plot: *spez. therm Energie pro* m^2 *[BST]* after cleanup

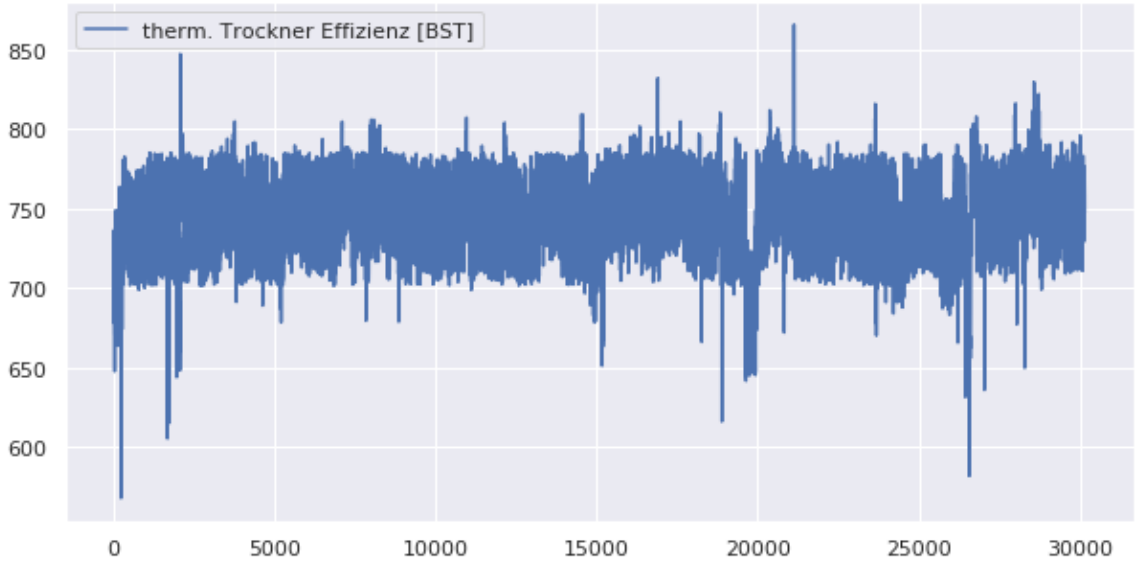


Figure A.32: Line Plot: *therm. Trockner Effizienz [BST]* after cleanup

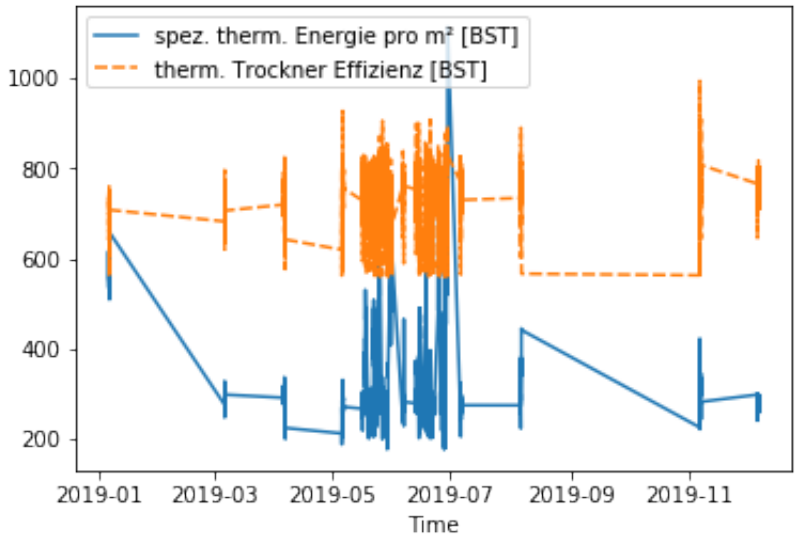


Figure A.33: Line Plot: *therm. Trockner Effizienz [BST]* merged with *spez. therm Energie pro m^2 [BST]* and scaled to compare curves (all)

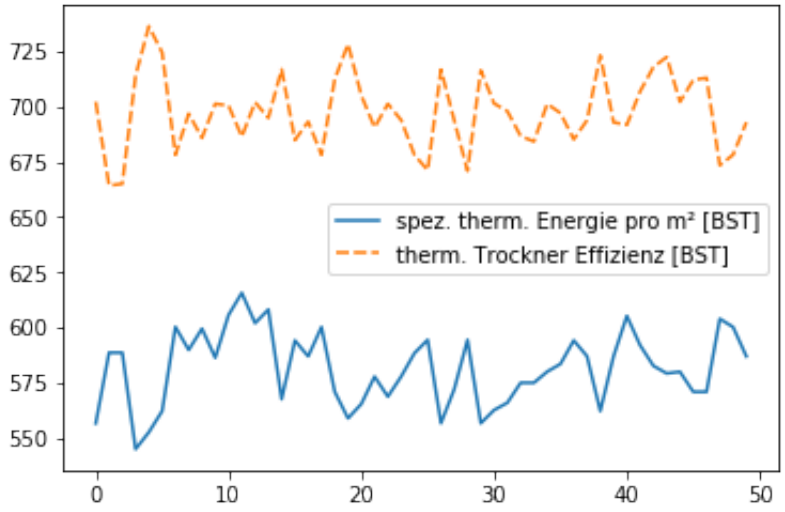


Figure A.34: Line Plot: *therm. Trockner Effizienz [BST]* merged with *spez. therm Energie pro m^2 [BST]* and scaled to compare curves (sample size of 50)

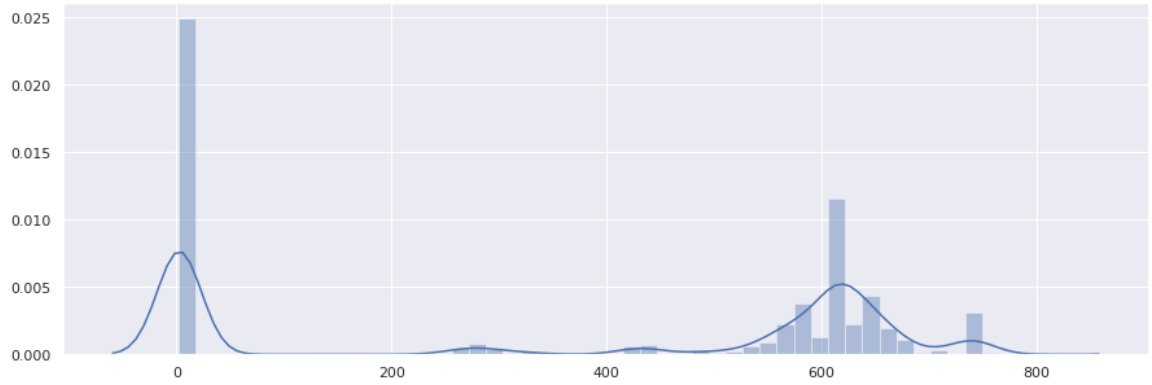


Figure A.35: Distribution Plot: *Gipsverbrauch aktuell kg/min*

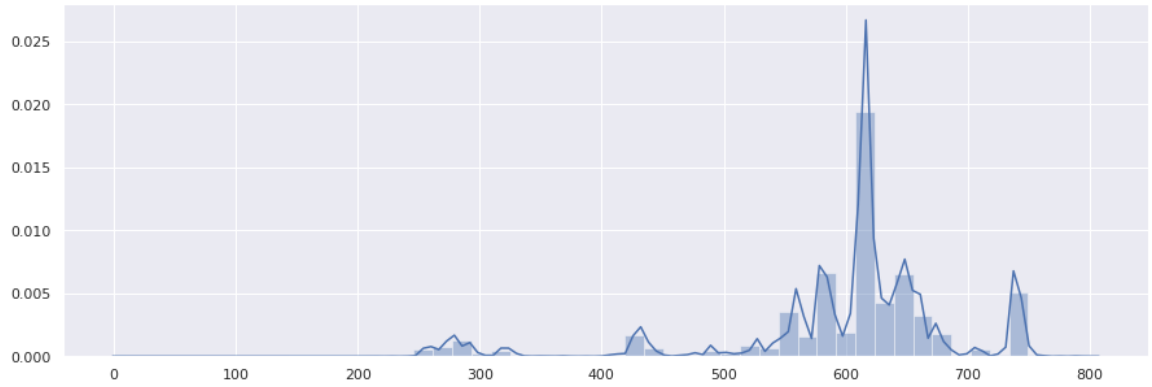


Figure A.36: Distribution Plot: *Gipsverbrauch aktuell kg/min* without zeros

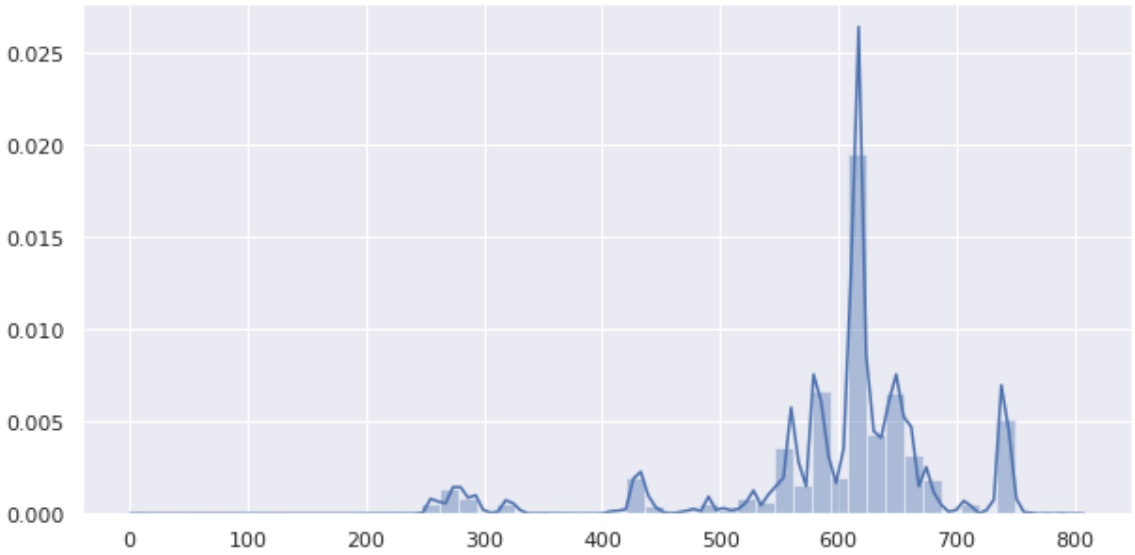


Figure A.37: Distribution Plot: *Gipsverbrauch aktuell kg/min* without zeros and outliers

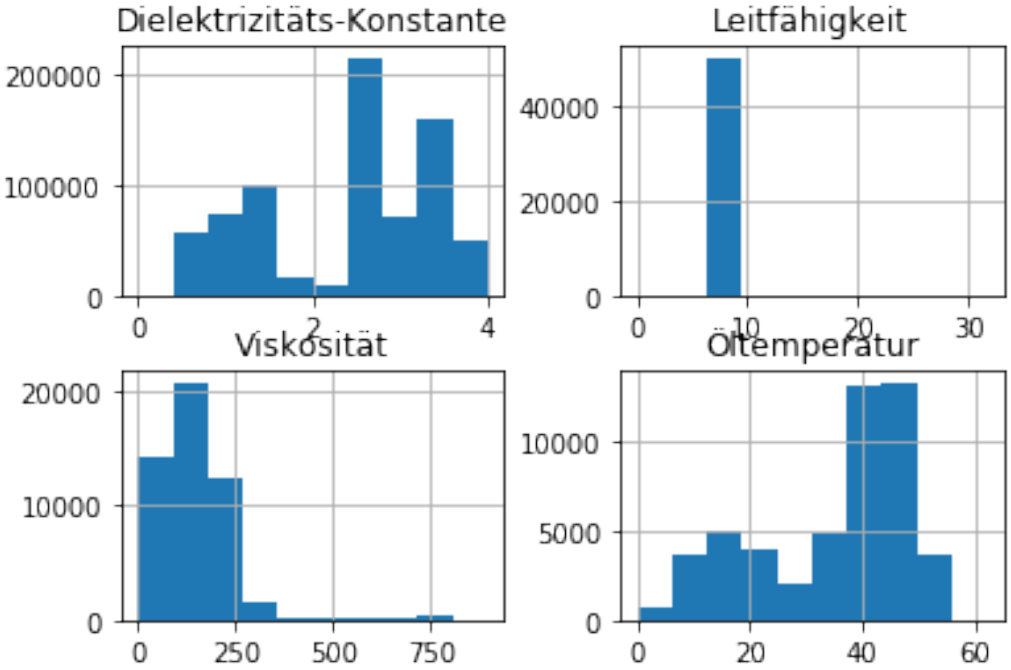


Figure A.38: Histogram: *trocknergetriebe*

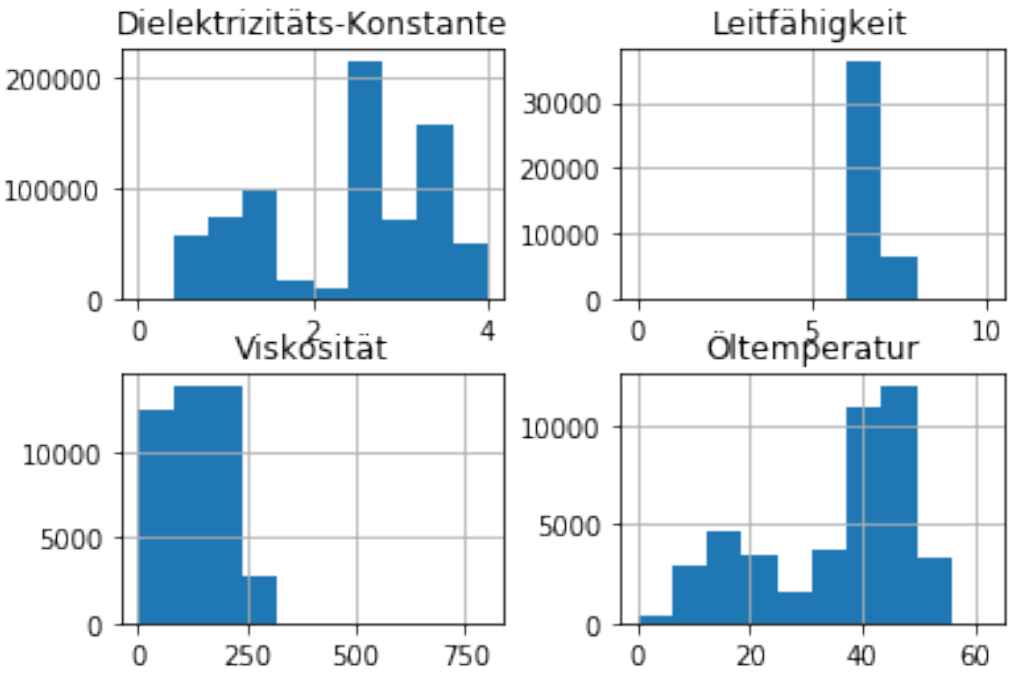


Figure A.39: Histogram: *trocknergetriebe* without outliers

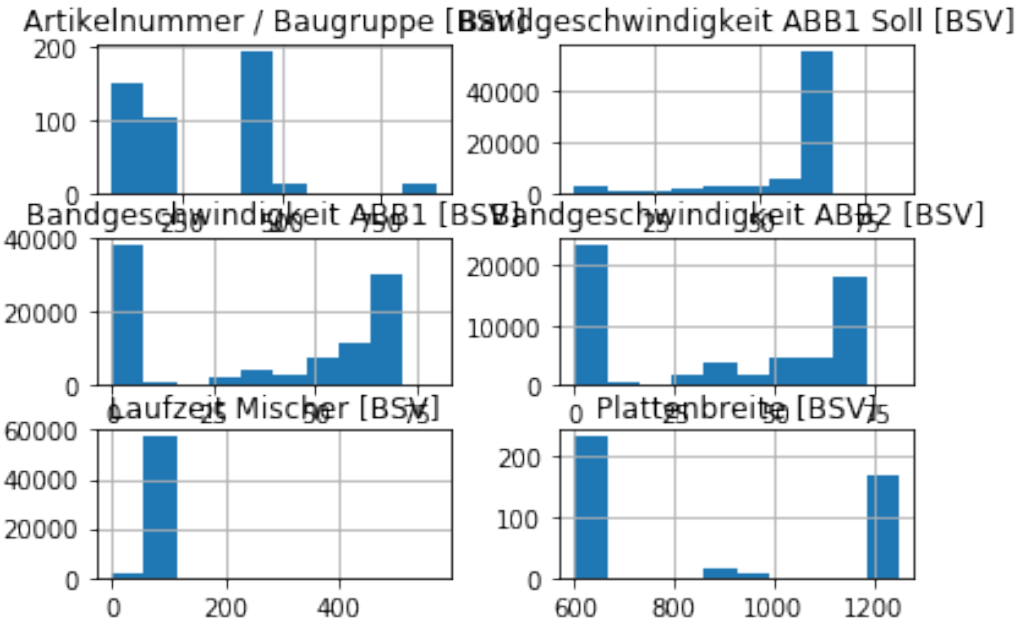


Figure A.40: Histogram: *vorbereitungsstation*

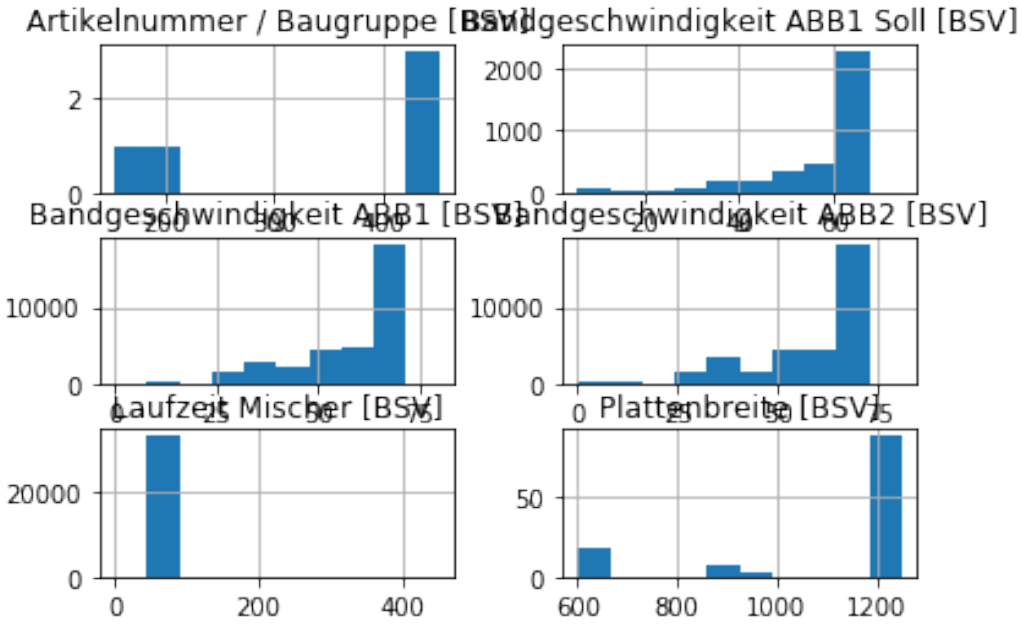


Figure A.41: Histogram: *vorbereitungsstation* without zeros for *Bandgeschwindigkeit ABB1 [BSV]* and *Bandgeschwindigkeit ABB2 [BSV]*

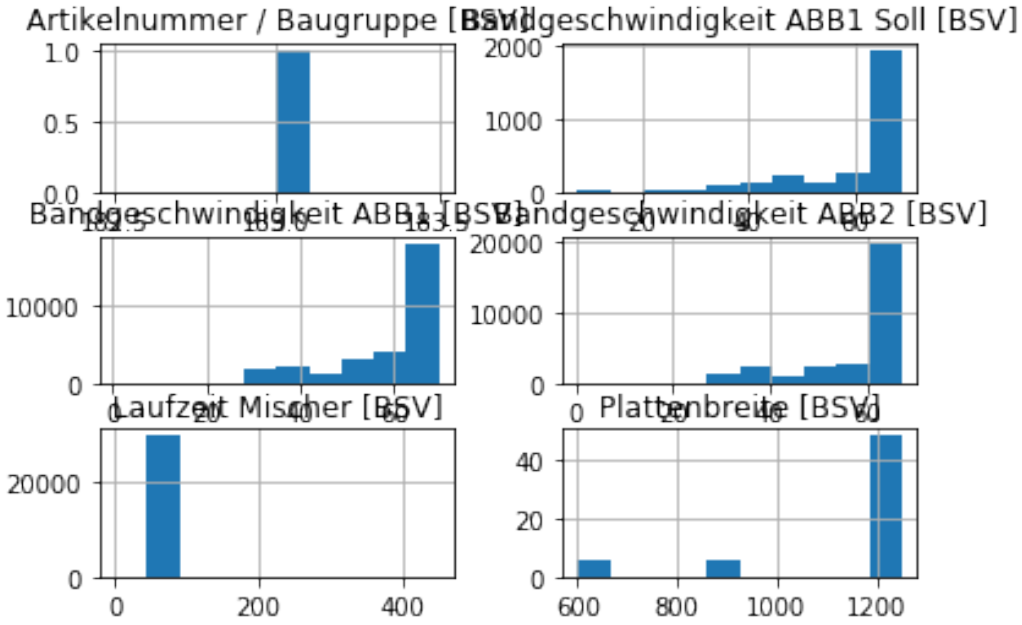


Figure A.42: Histogram: *vorbereitungsstation* without zeros and outliers

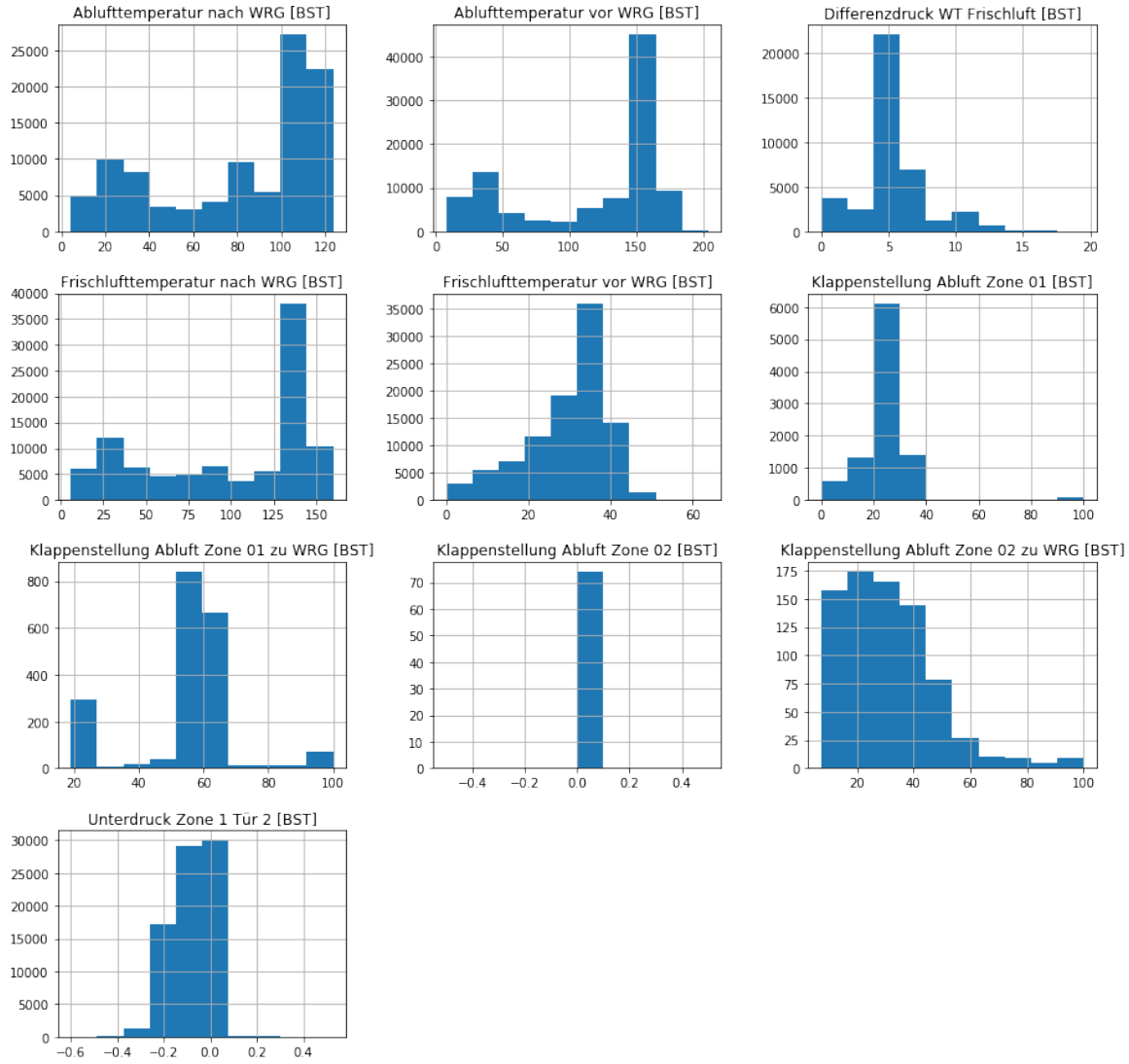


Figure A.43: Histogram: *waerme_rueckgewinnung*

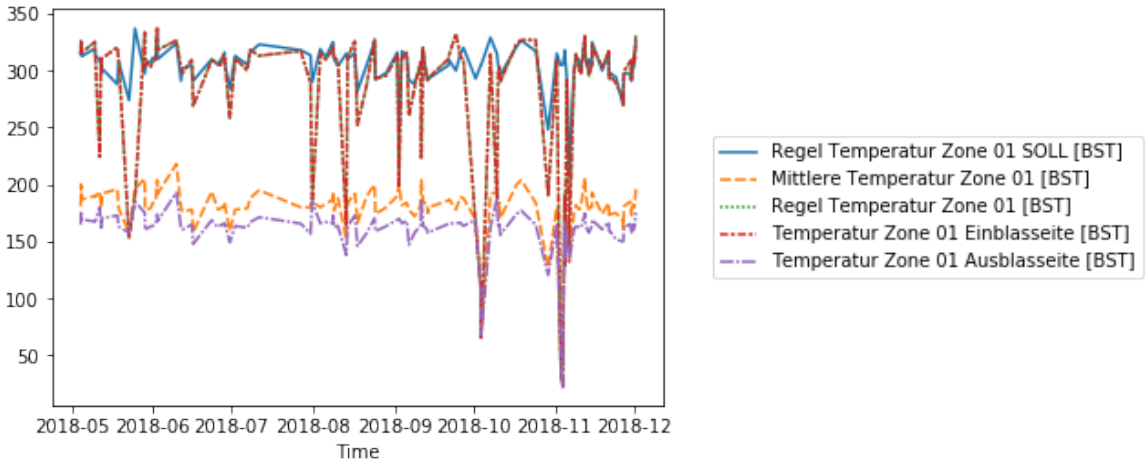


Figure A.44: Line Plot: *zone01* Temperature Attributes (sample size of 150)

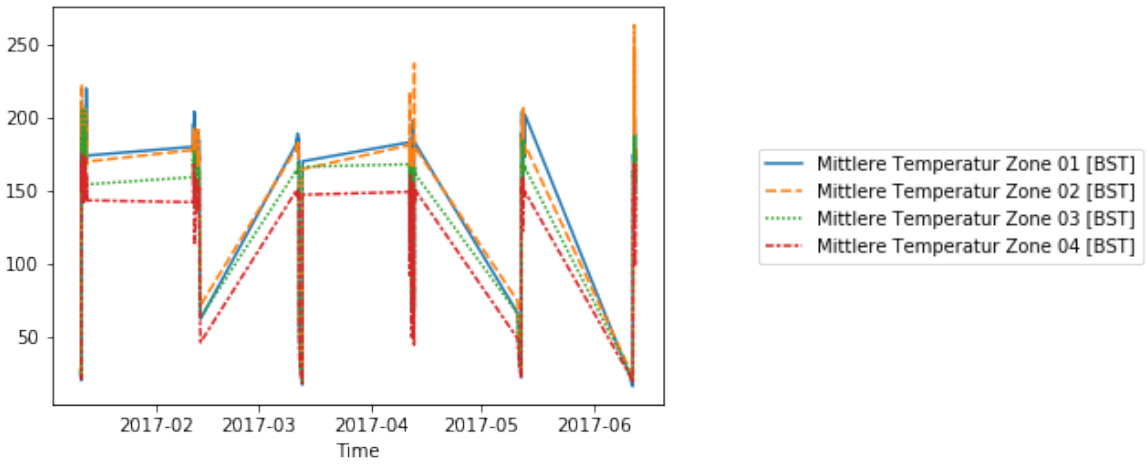


Figure A.45: Line Plot: *Mittlere Temperatur* Attributes of *zone01*, *zone02*, *zone03* and *zone04* (sample size of 1000)

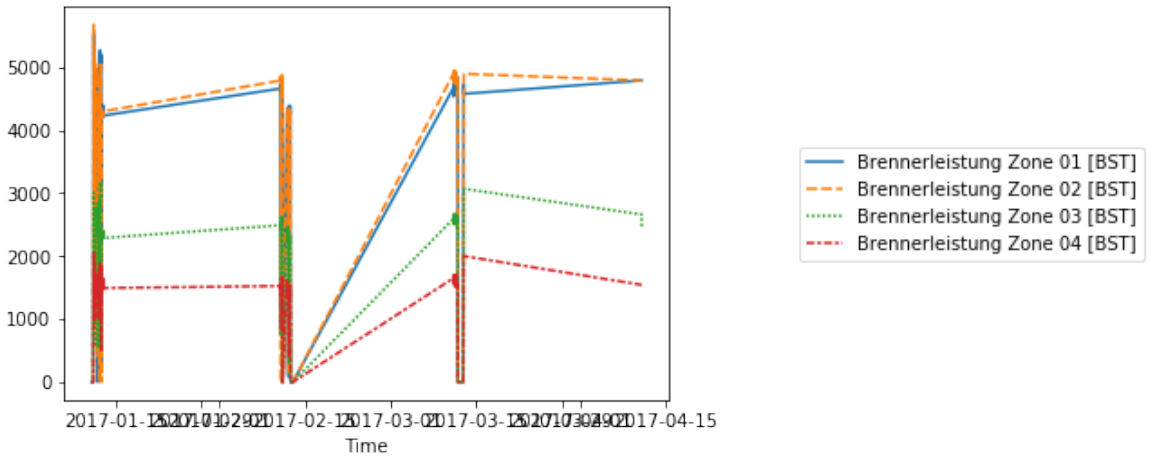


Figure A.46: Line Plot: *Brennerleistung* Attributes of *zone01*, *zone02*, *zone03* and *zone04* (sample size of 500)

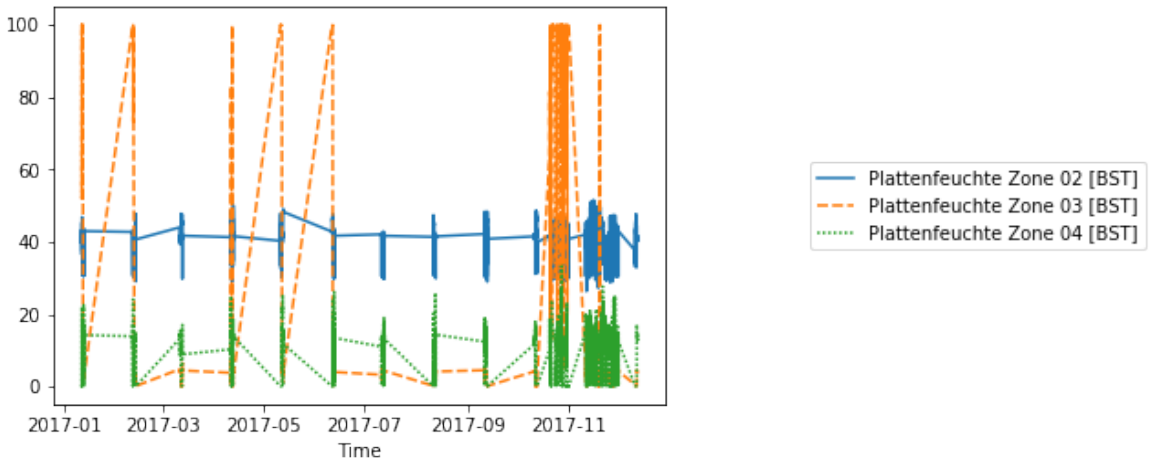


Figure A.47: Line Plot: *Plattenfeuchte* Attributes of *zone02*, *zone03* and *zone04* (sample size of 500)

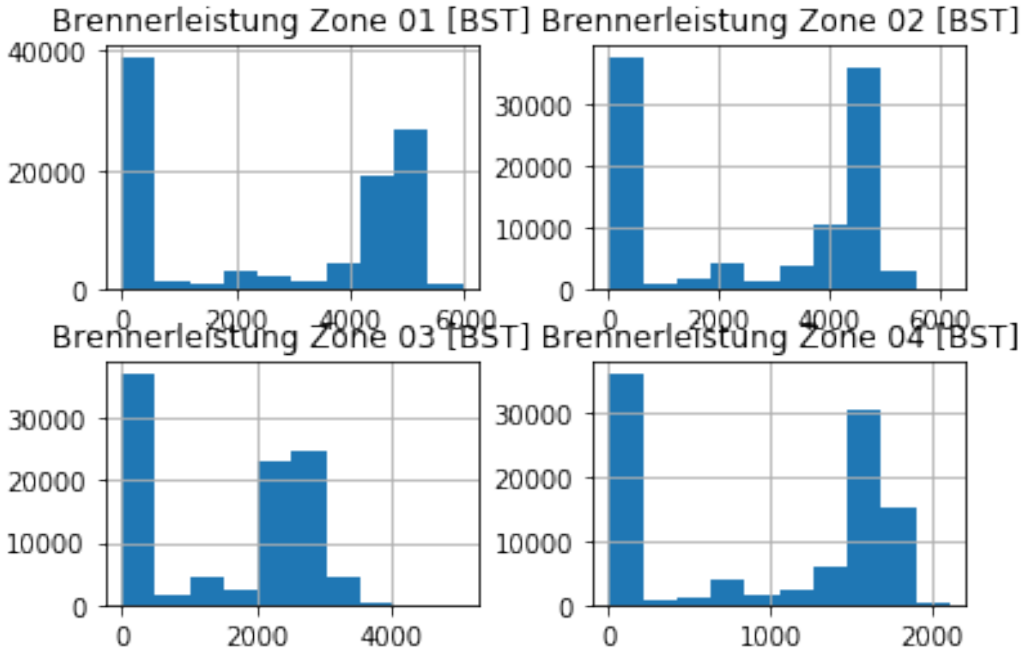


Figure A.48: Histogram: *Brennerleistung* Attributes of *zone01*, *zone02*, *zone03* and *zone04*

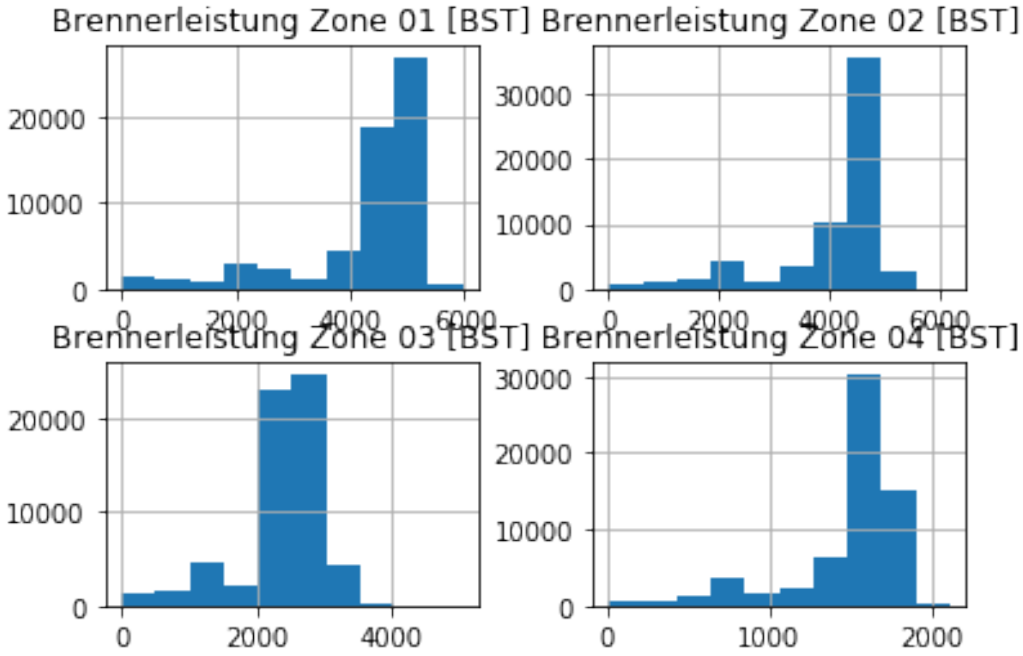


Figure A.49: Histogram: *Brennerleistung* Attributes of *zone01*, *zone02*, *zone03* and *zone04* without zeros

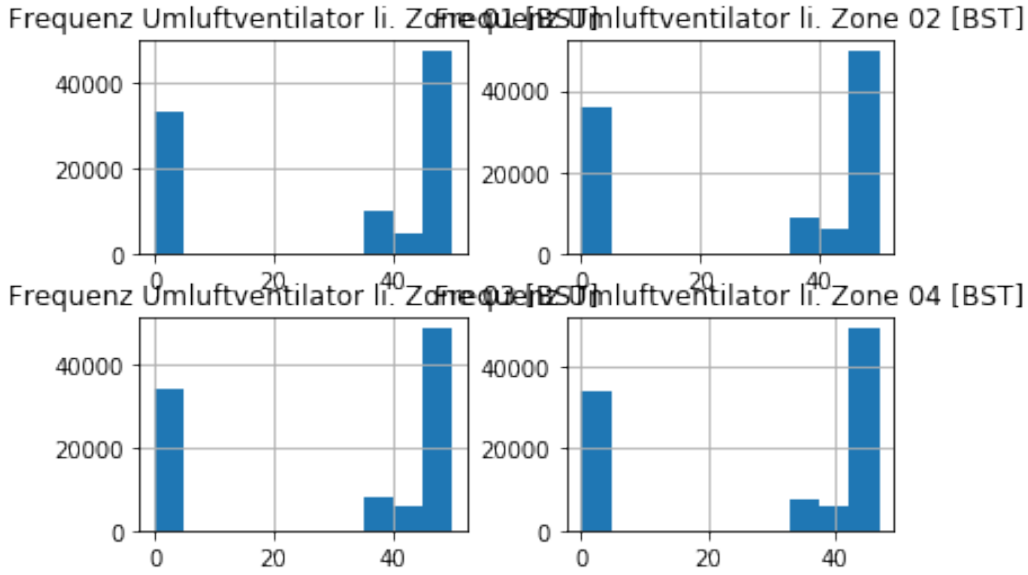


Figure A.50: Histogram: *Frequenz Umluftventilator li.* Attributes of *zone01*, *zone02*, *zone03* and *zone04*

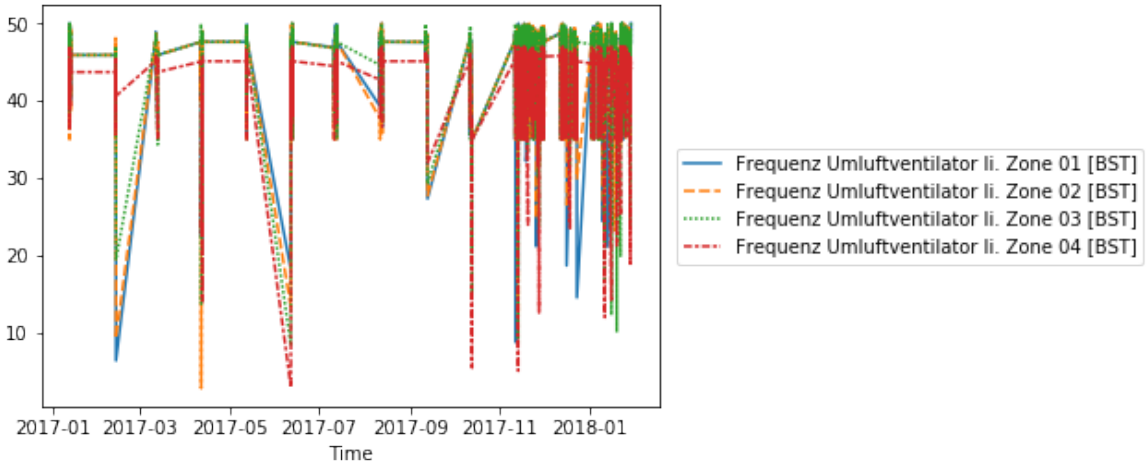


Figure A.51: Line Plot: *Frequenz Umluftventilator li.* Attributes of *zone01*, *zone02*, *zone03* and *zone04* (sample size of 5000)

A.3 Data Preparation

A.3.1 Feature Selection

The tables in A.19, A.20, A.21 and A.22 below summarize the findings of the business understanding and data understanding phase in regards to the data preparation phase. Each attribute exported is analyzed and characteristics or actions are documented. These are either important for further steps or should be performed on the attribute during data preparation. In the following, the descriptions for the abbreviations used in the table are explained.

Abbreviation (Column)	**Description**
ID	Index in the generated column lookup table
CO	Name of the attribute
P	Attribute should be propagated
O	Outlier detection should be performed
CR	Attribute should be cropped, range in brackets
D	Attribute should be dropped
CA	Attribute is categorical
T	Attribute is target
NV	Attribute has no Values
LS	Attribute has a low value saturation

Abbreviation (Cell)	**Decision Based On**
d	Domain Knowledge
x	Data Description and Exploration Report
v	Exploratory Data Analysis (Visualization)

ID	CO	P	O	CR	D	CA	T	NV	LS
0	Schmierung Tisch 1					x			
1	Summe Stromverbrauch Abnahme				x			x	
2	Trockenausschuss [BSA]				x			x	
3	Restfeuchte Platten [BSA]								
4	Anzahl Pakete Abnahme		x						

Table A.19: Feature Table Part I

ID	CO	P	O	CR	D	CA	T	NV	LS
5	Doppler 2 Bänder		x						x
6	Laufzeit Abnahme				x			x	
7	Knauf Stadtoldendorf.02...			x		x	x		x
8	Ende								
9	Gesamtleistung Brenner [BST]		v	v (>0)					
10	Heizwert Erdgas Hu				x				x
11	Laufzeit Trockner [BST]				x			x	
12	Nassausschuss [BST]				x			x	
13	Summe Gasverbrauch Trockner [BST]				x			x	
14	Summe Stromverbrauch Trockner [BST]				x			x	
15	Trocknerleistung		v	v (>0)					
16	spez. therm. Energie pro m^2 [BST]		v	v (>0)			d		x
17	therm. Trockner Effizienz [BST]		v	d (2000>x<560)			d		x
18	zu verdampfendes Wasser pro m^2 [BST]			d (>= 0)					x
19	Gipsverbrauch aktuell kg/min		v	v (>0)					
20	Plattenbreite in m [BSV]				d				x
21	Reinheitsgrad Gips				x				x
22	Öltemperatur	x							x
23	Viskosität				v				
24	Dielektrizitäts-Konstante								
25	Leitfähigkeit				v				
26	Laufzeit Mischer [BSV]								x
27	Laufmeter Bandstraße				x			x	
28	Summe Stromverbrauch Vorbereitungssation				x			x	
29	Plattenbreite [BSV]	x				v	x		
30	Bandgeschwindigkeit ABB1 [BSV]			d (>0)					x
31	Artikelnummer / Baugruppe [BSV]	x			x		x		x
32	Bandgeschwindigkeit ABB2 [BSV]			d (>0)					

Table A.20: Feature Table Part II

ID	CO	P	O	CR	D	CA	T	NV	LS
33	Bandgeschwindigkeit ABB1 Soll [BSV]				d				
34	Klappenstellung Abluft Zone 02 [BST]	x							x
35	Klappenstellung Abluft Zone 01 zu WRG SOLL [BST]				d				
36	Klappenstellung Abluft Zone 01 zu WRG [BST]	x							x
37	Klappenstellung Abluft Zone 02 SOLL [BST]				d				
38	Unterdruck Zone 1 Tür 2 [BST]								
39	Klappenstellung Abluft Zone 02 zu WRG SOLL [BST]				d				
40	Klappenstellung Abluft Zone 02 zu WRG [BST]	x							x
42	Differenzdruck WT Frischluft [BST]								
43	Ablufttemperatur vor WRG [BST]				v				x
44	Frischlufttemperatur nach WRG [BST]	x							x
45	Klappenstellung Abluft Zone 01 SOLL [BST]				d				
46	Klappenstellung Abluft Zone 01 [BST]	x							x
47	Frischlufttemperatur vor WRG [BST]	x							x
48	Regel Temperatur Zone 01 SOLL [BST]				d				
49	Regel Temperatur Zone 01 [BST]				v				
50	Temperatur Zone 01 Einblasseite [BST]				v				
51	Temperatur Zone 01 Ausblasseite [BST]				v				
52	Mittlere Temperatur Zone 01 [BST]								
53	Feuchtebeladung Luft Zone 01 [BST]	x							x
54	Brennerleistung Zone 01 [BST]	x		v (>0)					x
55	Frequenz Umluftventilator re. Zone 01 [BST]	x							x
56	Frequenz Umluftventilator li. Zone 01 [BST]				d				
57	Regel Temperatur Zone 02 [BST]				v				
58	Plattenfeuchte Zone 02 [BST]								
59	Regel Temperatur Zone 02 SOLL [BST]				d				
60	Temperatur Zone 02 Einblasseite [BST]				v				
61	Temperatur Zone 02 Ausblasseite [BST]				v				

Table A.21: Feature Table Part III

ID	CO	P	O	CR	D	CA	T	NV	LS
62	Feuchtebeladung Luft Zone 02 [BST]								
63	Brennerleistung Zone 02 [BST]				v				
64	Frequenz Umluftventilator li. Zone 02 [BST]				d				
65	Mittlere Temperatur Zone 02 [BST]				v				
66	Frequenz Umluftventilator re. Zone 02 [BST]				v				
67	Regel Temperatur Zone 03 [BST]				v				
68	Plattenfeuchte Zone 03 [BST]								
69	Regel Temperatur Zone 03 SOLL [BST]				d				
70	Temperatur Zone 03 Einblasseite [BST]				v				
71	Temperatur Zone 03 Ausblasseite [BST]				v				
72	Feuchtebeladung Luft Zone 03 [BST]								
73	Brennerleistung Zone 03 [BST]				v				
74	Frequenz Umluftventilator li. Zone 03 [BST]				d				
75	Mittlere Temperatur Zone 03 [BST]				v				
76	Frequenz Umluftventilator re. Zone 03 [BST]				v				
77	Regel Temperatur Zone 04 SOLL [BST]				d				
78	Regel Temperatur Zone 04 [BST]				v				
79	Temperatur Zone 04 Einblasseite [BST]				v				
80	Temperatur Zone 04 Ausblasseite [BST]				v				
81	Plattenfeuchte Zone 04 [BST]								
82	Frequenz Umluftventilator li. Zone 04 [BST]				d				
83	Brennerleistung Zone 04 [BST]				v				
84	Mittlere Temperatur Zone 04 [BST]								
85	Frequenz Umluftventilator re. Zone 04 [BST]				v				

Table A.22: Feature Table Part IV

A.3.2 Parallel Processing

Because again transformations and changes are performed on many files, the *billiard* package is used to parallelize these. Also a column list is created to keep it as a lookup table for the feature names.

To finish the setup, all the files are read into memory. This is done by using the *Pool* class of *billiard* to read the files into a list of dataframes that can be iterated to perform actions on all of them. By calling the *map*-method of a *Pool* object and passing a function and an iterable, the function is executed on every item of the iterable in parallel. The number of maximum processes created can be defined when creating the *Pool* object. To calculate this number, a function is implemented taking into account the number of threads of the system and a reference which e.g. can be the number of files to be read in. The algorithm then decides how many threads to allocate for the task. If no reference is provided, 90% of the threads (rounded off) are used as the number of processes to still leave capacity open for background tasks.

A.4 Modeling

A.4.1 Library Selection

To help implement the selected algorithms, several libraries can be used to simplify the process by adding a layer of abstraction. Most libraries offer optimized implementations of the most commonly used algorithms and transformation. In the area of data science and data analysis two programming languages are commonly used: *Python* and *R*. Due to personal preferences and selection of the libraries, python will be the programming language of choice in this project and the libraries described are all developed, though not exclusively, for the language.

The first library that is used is called *scikit-learn*. It builds on well-known python packages like *NumPy*, *SciPy* and *matplotlib*.[cf. 45] These underlying packages offer various functionality like support for n-dimensional array objects and linear algebra, providing a collection of software for mathematics, science and engineering or implement options to plot data in 2D.[cf. 45, cf. 46, cf. 47] Scikit-learn provides implementations of algorithms regarding classification, regression, clustering, dimensionality reduction, model selection and preprocessing.[cf. 45] The library is licensed under the open source license BSD, which also allows commercial use.[cf. 45] To sum it up, scikit-learn covers most of the needed algorithms and also provides useful tools for data preprocessing like data cleaning, feature selection and feature engineering.

Data is read and handled using *pandas*. This library provides a data structure called *DataFrame* that allows the user to easily manipulate and analyze numerical tables and time series.[cf. 48]

For the gradient boosting algorithm, the XGBoost library is chosen as it promises a highly efficient implementation. During the training process an attempt should be made to verify whether or not a better model performance can be achieved and if it lives up to the expectations regarding training time and parallelization.[cf. 15]

Because scikit-learn does not provide a lot of functionality regarding ANNs, *Keras* is used to cover this area. The package uses the *TensorFlow* backend and adds another layer of abstraction to it, making the process of setting up and training neural networks more comfortable and accessible.[cf. 49] It also provides a wrapper to include sequential Keras models into the scikit-learn workflow by offering methods expected from a scikit-learn estimator.[cf. 50]

As mentioned above, Keras uses the TensorFlow backend. "TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization for the purposes of conducting machine learning and deep neural networks research."[51] It is also very versatile and can be used on different platforms by utilizing e.g. *TensorFlow.js* for a web environment or *TensorFlow Lite* for mobile and IoT purposes.[cf. 52]

A.4.2 Class Selection

Implementing the machine learning algorithms from scratch does require a lot of time and deep insight into the mathematics behind them. As mentioned earlier, scikit-learn offers optimized implementations of the most important algorithms and provides a high level API for easy use. The different classes used in this project are listed in A.23. Scikit-learn's API can be divided into two main principles, *estimators* and *transformers*.[cf. 6, p. 61] Estimators provide a *fit*-method that takes a dataset as parameter and trains the model on this.[cf. 6, p. 61] By then calling the *predict*-method, predictions can be made by passing another dataset. A transformer on the other hand transform datasets.[cf. 6, p. 61] The transformer can learn from parameters of the dataset by calling the *fit*-method and passing in the dataset.[cf. 6, p. 61] By then using the *transform*-method, the transformer takes the data and applies those parameters to it, returning the transformed data. The regressors used in the following training process can be identified as an estimator. The class used to scale the data is of type transformer. To run the pipeline, the *fit*- and *predict*-methods that are provided by the estimators are also provided by the pipeline. In the majority of cases, the constructor of those classes can be called to create an object which is then fed into a pipeline. For some this approach is not sufficient and has therefore be described in the

following. Also some models incorporate stochastic techniques, depending on random interactions. To ensure a reproducibility of the runs, the *random_state* parameter is always set to *42* if possible.

The first algorithm that is implemented differently is polynomial regression. As mentioned in the theoretical foundations, new features are created which are then used to train a model. In scikit-learn one can make use of the *PolynomialFeatures* class. This class is of type transformer and therefore takes the dataset and some hyperparameters, returning a transformed dataset that includes the new engineered features. The dataset is then used to train different models. In this thesis, a standard linear regression is used as a starting point and compared with ridge as well as lasso regression. Because many new features are created and linear regression with regularization does perform an automatic feature selection, the latter two are expected to perform better. To combine the different parts of the process, the transformer object can be included into the pipeline.

Another algorithm that requires a more complex implementation is the gradient boosting variant of random forest. As mentioned earlier, *XGBoost* is the implementation of choice. Because it is not a part of the scikit-learn library and does not support multi output regression, it has to be wrapped in a *MultiOutputRegressor* object. The underlying class is provided by scikit-learn and trains a single regressor for each output variable, later wrapping them into a single regressor that is capable of performing multi output regression. This process is possible, because the *XGBRegressor* class of the XGBoost library is used, wrapping the original implementation into a scikit-learn compatible estimator.[cf. 53]

To include the ANNs and DNNs implemented with Keras, a scikit-learn wrapper provided by the package itself is used to wrap the model into a *KerasRegressor* object. This object behaves like a scikit-learn estimator, offering e.g. a *fit*- and *predict*-method and is therefore compatible with the library and its functions. This way a pipeline can be created and a randomized search can be conducted on Keras models.

Table A.23: Classes Used to Implement the Algorithms

Algorithm	Class	Library
Linear Regression	sklearn.linear_model.LinearRegression	Scikit-Learn
Ridge Regression	sklearn.linear_model.Ridge	Scikit-Learn
Lasso Regression	sklearn.linear_model.Lasso	Scikit-Learn
Elastic net	sklearn.linear_model.ElasticNet	Scikit-Learn
Support Vector Machine	sklearn.svm.SVR	Scikit-Learn
Polynomial Regression	sklearn.preprocessing.PolynomialFeatures, sklearn.linear_model.LinearRegression	Scikit-Learn
Random Forest Regression	sklearn.ensemble.RandomForestRegressor	Scikit-Learn
Gradient Boosting	sklearn.multioutput.MultiOutputRegressor, xgboost.XGBRegressor	Scikit-Learn
ANN	keras.models.Sequential, keras.wrappers.scikit_learn.KerasRegressor	Keras

A.4.3 Randomized Search Parameter Grids

```
{
'reg__fit_intercept': [True, False],
'reg__normalize': [True, False]
}
```

Model Parameters A.1: Parameter Grid: LinearRegression

```
{
'reg__alpha': [1e-8, 1e-4, 1e-3, 1e-2, 1, 5, 10],
'reg__fit_intercept': [True, False],
'reg__solver': ['svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']
}
```

Model Parameters A.2: Parameter Grid: Ridge

```
{
'reg__alpha': [1e-8, 1e-4, 1e-3, 1e-2, 1, 5, 10],
'reg__fit_intercept': [True, False],
'reg__tol': [1e-8, 1e-4, 1e-3, 1e-2, 1, 5, 10],
'reg__positive': [True, False],
'reg__selection': ['cyclic', 'random']
}
```

Model Parameters A.3: Parameter Grid: Lasso

```
{
'reg__alpha': [1e-8, 1e-4, 1e-3, 1e-2, 1, 5, 10],
'reg__l1_ratio': [1e-8, 1e-4, 1e-3, 1e-2, 1, 5, 10],
'reg__fit_intercept': [True, False],
'reg__tol': [1e-8, 1e-4, 1e-3, 1e-2, 1, 5, 10],
'reg__positive': [True, False],
'reg__selection': ['cyclic', 'random']
}
```

Model Parameters A.4: Parameter Grid: Elastic net

```
{
'poly__interaction_only': [True, False],
'poly__include_bias': [True, False],
'poly__order': ['C', 'F'],
...
+ paramteres depending on algorithm used for prediction
}
```

Model Parameters A.5: Parameter Grid: PolynomialFeatures

```
{
'mop__estimator__gamma':['auto', 'scale'],
'mop__estimator__shrinking': [True, False],
'mop__estimator__tol':[1e-4, 1e-3, 1e-2, 1, 5],
'mop__estimator__C':[1e-4, 1e-3, 1e-2, 1, 5],
'mop__estimator__epsilon':[1e-4, 1e-3, 1e-2, 1, 5]
}
```

Model Parameters A.6: Parameter Grid: SVR

```
{
'reg__criterion': ['mse', 'mae'],
'reg__max_depth': [None, 2, 5, 10, 15],
'reg__min_samples_split': [2, 3, 5, 10],
'reg__min_samples_leaf': [1, 2, 3],
'reg__max_features': ['auto', 'sqrt', 'log2'],
'reg__bootstrap': [True, False],
}
```

Model Parameters A.7: Parameter Grid: RandomForestRegressor

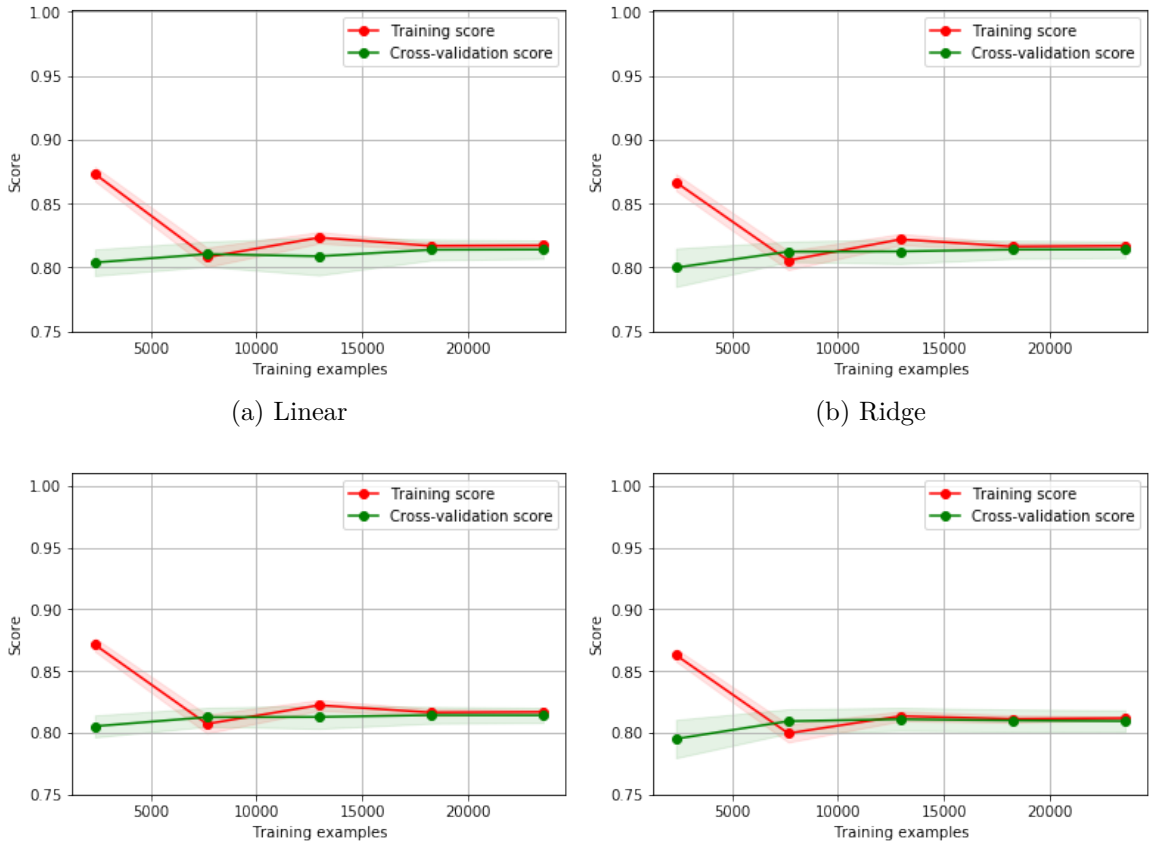
```
{
'mop__estimator__n_estimators': [10, 25, 50, 100, 250, 500, 750],
'mop__estimator__learning_rate': [.0001, .001, .01, .1, .2, .3, .5, 1],
'mop__estimator__max_depth': [2, 5, 6, 10, 15],
'mop__estimator__booster': ['gbtree', 'gblinear', 'dart'],
'mop__estimator__reg_lambda': [.1, .3, .5, .7, 1],
'mop__estimator__reg_alpha': [.1, .3, .5, .7, 1]
}
```

Model Parameters A.8: Parameter Grid: XGBRegressor

```
{
'reg__optimizer': ['adam', 'adagrad', 'rmsprop', 'sgd'],
'reg__epochs': [10, 20, 30, 50, 100],
'reg__batch_size': [4, 8, 16, 24, 32, 64],
'reg__neurons': [4, 8, 16, 25, 32]
'reg__weight_constraint': [1, 2, 3, 4, 5]
}
```

Model Parameters A.9: Parameter Grid: Keras Model

A.4.4 Learning Curves



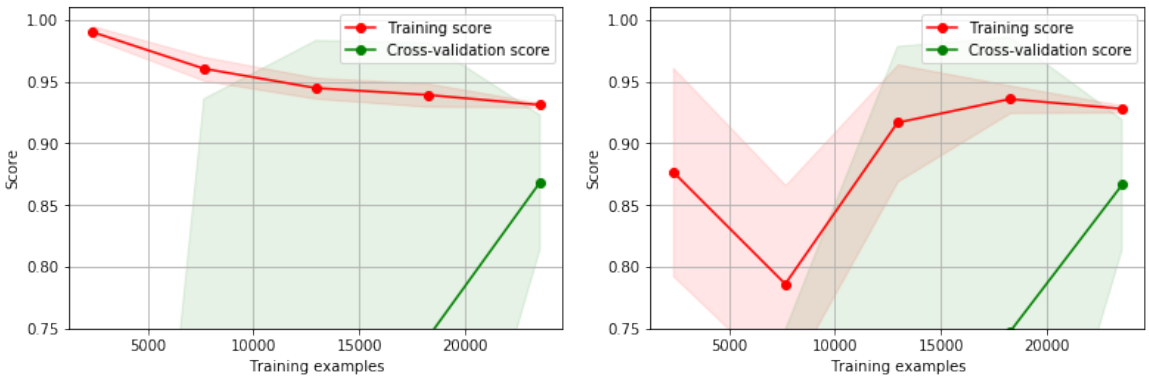
(a) Linear

(b) Ridge

(c) Lasso

(d) Elasicnet

Figure A.52: Learning Curves: Linear Regression + Regularization



(a) Linear (b) Ridge

Figure A.53: *Learning Curves Polynomial Features + Linear Regression + Regularization*

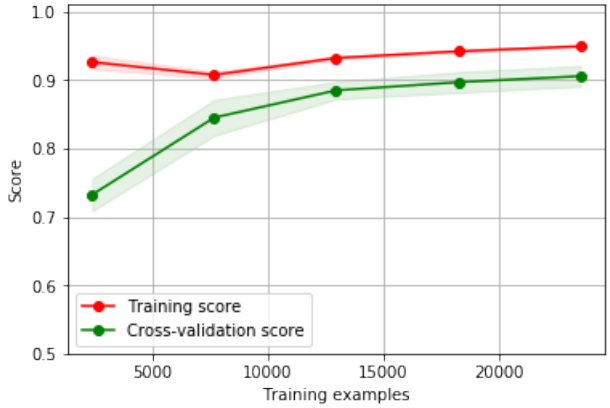
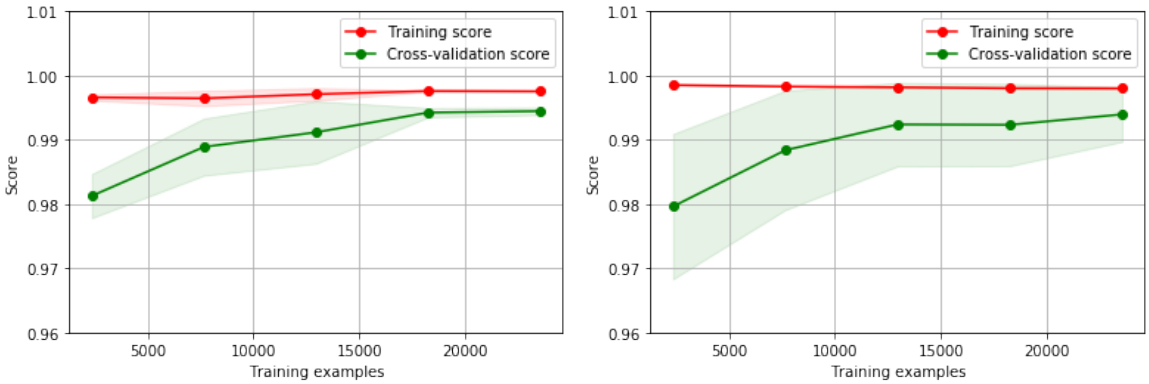


Figure A.54: Learning Curve: SVR



(a) Random Forest (b) XGBoost

Figure A.55: Learning Curves: Random Forest

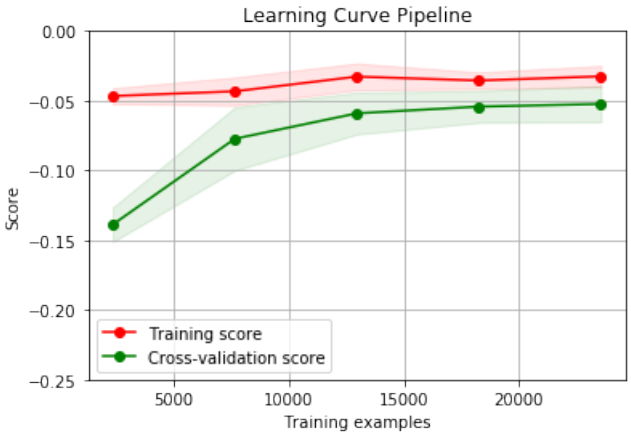


Figure A.56: Learning Curve: Keras Model

A.4.5 Best Estimators

```
Pipeline(
memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
 with_std=True)),
('reg', LinearRegression(copy_X=True,
 fit_intercept=False,
 n_jobs=None,
 normalize=True))],
verbose=False)
```

Model Parameters A.10: Best Estimator: LinearRegression

```
Pipeline(memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
  with_std=True)),
('reg', Ridge(alpha=1e-08,
 copy_X=True,
 fit_intercept=False,
 max_iter=None,
 normalize=False,
 random_state=42,
 solver='sag',
 tol=0.001))],
verbose=False)
```

Model Parameters A.11: Best Estimator: Ridge

```
Pipeline(memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
 with_std=True)),
('reg', Lasso(alpha=1e-08,
 copy_X=True,
 fit_intercept=False,
 max_iter=1000,
 normalize=False,
 positive=False,
 precompute=False,
 random_state=42,
 selection='cyclic',
 tol=1e-08,
 warm_start=False))],
verbose=False)
```

Model Parameters A.12: Best Estimator: Lasso

```
Pipeline(memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
 with_std=True)),
('reg', ElasticNet(alpha=1e-08,
 copy_X=True,
 fit_intercept=False,
 l1_ratio=1,
 max_iter=1000,
 normalize=False,
 positive=False,
 precompute=False,
 random_state=42,
 selection='random',
 tol=0.0001,
 warm_start=False))],
verbose=False)}
```

Model Parameters A.13: Best Estimator: ElasticNet

```
Pipeline(memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
 with_std=True)),
('reg', RandomForestRegressor(bootstrap=True,
 criterion='mse',
 max_depth=15, max_features='auto',
 max_leaf_nodes=None,
 min_impurity_decrease=0.0,
 min_impurity_split=None,
 min_samples_leaf=1,
 min_samples_split=3,
 min_weight_fraction_leaf=0.0,
 n_estimators=10,
 n_jobs=-1,
 oob_score=False,
 random_state=42,
 verbose=0,
 warm_start=False))],
verbose=False)
```

Model Parameters A.14: Best Estimator: RandomForestRegressor

```
Pipeline(memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
  with_std=True)),
('mop', MultiOutputRegressor(estimator=XGBRegressor(base_score=0.5,
 booster='gbtree',
 colsample_bylevel=1,
 colsample_bynode=1,
 colsample_bytree=1,
 gamma=0,
 importance_type='gain',
 learning_rate=0.5,
 max_delta_step=0,
 max_depth=2,
 min_child_weight=1,
 missing=nan,
 n_estimators=500,
 n_jobs=-1,
 nthread=None,
 objective='reg:linear',
 random_state=42,
 reg_alpha=0.7,
 reg_lambda=0.1,
 scale_pos_weight=1,
 seed=None,
 silent=None,
 subsample=1,
 verbosity=1),
 n_jobs=None))],
verbose=False)
```

Model Parameters A.15: Best Estimator: XGBRegressor

```
Pipeline(memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
 with_std=True)),
('poly', PolynomialFeatures(degree=2,
 include_bias=True,
 interaction_only=True,
 order='C')),
('reg', LinearRegression(copy_X=True,
 fit_intercept=True,
 n_jobs=None,
 normalize=True))],
verbose=False)
```

Model Parameters A.16: Best Estimator: PolynomialFeatures + LinearRegression

```
Pipeline(memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
 with_std=True)),
('poly', PolynomialFeatures(degree=2,
 include_bias=True,
 interaction_only=True,
 order='C')),
('reg', Ridge(alpha=10,
 copy_X=True,
 fit_intercept=True,
 max_iter=None,
 normalize=False,
 random_state=None,
 solver='svd',
 tol=0.001))],
verbose=False)
```

Model Parameters A.17: Best Estimator: PolynomialFeatures + Ridge

```
Pipeline(memory=None,
steps=[
('scaler', StandardScaler(copy=True,
 with_mean=True,
 with_std=True)),
('mop', MultiOutputRegressor(estimator=SVR(C=5,
 cache_size=200,
 coef0=0.0,
 degree=3,
 epsilon=0.01,
 gamma='scale',
 kernel='rbf',
 max_iter=-1,
 shrinking=True,
 tol=0.0001,
 verbose=2),
 n_jobs=-1))],
verbose=False)
```

Model Parameters A.18: Best Estimator: SVR

```
{
 'optimizer': 'adam',
 'neurons': 8,
 'epochs': 100,
 'batch_size': 16
}
```

Model Parameters A.19: Best Estimator: ANN

A.4.6 Impact Number of Neurons

Parameter	**ANN 1**	**ANN 2**	**ANN 3**
batch_size	32	32	32
epochs	100	100	100
early stopping after	13	13	10
validation_split	0.2	0.2	0.2
number of neurons	100	25	8
Score			
$R\hat{2}$ train	0.9180	0.9157	0.9365
$R\hat{2}$ test	0.8923	0.8910	0.9112

Table A.24: Results of Models Trained with Different Number of Neurons

A.4.7 Correlation Matrix

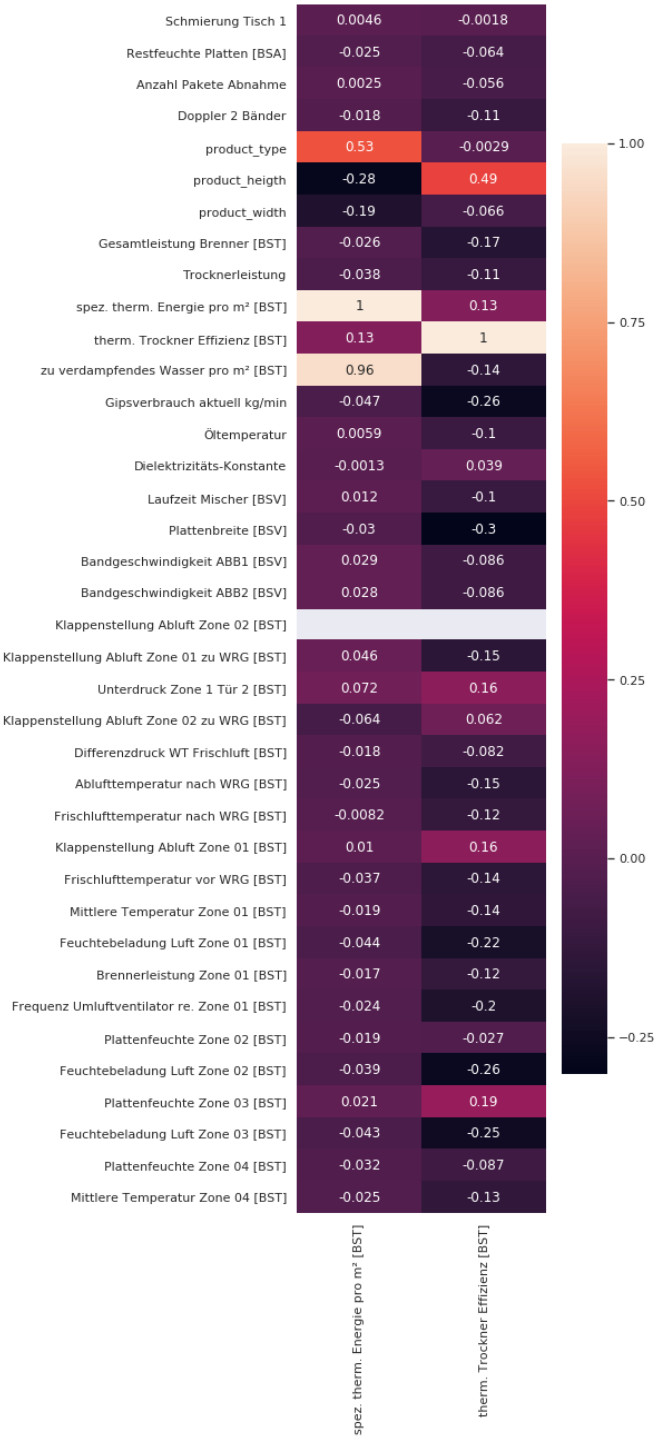


Figure A.57: Correlation Matrix

Declaration of honour

I hereby declare that I have written the present work with the topic: *Conception and Development of a Machine Learning Model for the Analysis of the Energy Consumption of a Plasterboard Dryer* independently and have not used any sources and tools other than those indicated. I also assure that the electronic version submitted is identical to the printed version.

Place, Date Marius Memmel