Marcel Öfele

A Hybrid Product Cost Estimation Approach Based on Outlier Removal and Machine Learning

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF

FAKULTÄT FÜR INFORMATIK

Computational Intelligence

A Hybrid Product Cost Estimation Approach Based on Outlier Removal and Machine Learning

Master Thesis

Marcel Öfele

25th January 2021

Supervisor:	Prof. Dr.-Ing. habil. Sanaz Mostaghim
Advisor:	Dr. Cristian Ramírez Atencia
Advisor:	Dr.-Ing. Christoph Steup
Advisor:	Dipl.-Phys. Wolfram Küter

Marcel Öfele: *A Hybrid Product Cost Estimation Approach Based on Outlier Removal and Machine Learning*
Otto-von-Guericke Universität
Computational Intelligence
Magdeburg, 25th January 2021.

Abstract

This work proposes a hybrid cost estimation approach for purchased products, which is designed to deal with the limited information and the uncertainty in the cost for products which are manufactured by suppliers. For this application scenario, a realistic cost estimation with common techniques is particularly difficult. The presented approach combats these difficulties by combining multiple different techniques to overcome their specific weaknesses. This is achieved by using historical cost data, filtering undesirable data points and training a machine learning model for the final cost estimation. At the heart of this approach are the newly developed outlier removal methods based on a simplified analytical estimation technique and a data envelopment analysis. These methods filter the historical data to remove parts whose costs are dominated by non-product related cost drivers.

A case study demonstrates the superiority of the proposed hybrid approach over a stand-alone machine learning model. Additionally, it shows that the outlier removal methods based on analytical techniques perform best in this approach. The case study further indicates that the usage of a data envelopment analysis for the filtering leads to less realistic cost estimates. Nevertheless, it can be a valid method for applications where cost efficiency takes utmost priority.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ANN	Artificial Neural Network
CDF	Cumulative Distribution Function
DEA	Data Envelopment Analysis
DMU	Decision Making Unit
EDF	Empirical Distribution Function
KS	Kolmogorov-Smirnov
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
PCA	Principal Component Analysis
RBF	Radial Basis Function
RMSE	Root Mean Squared Error
SGA	Selling, General and Administrative Expenses
SVM	Support-Vector Machine
SVR	Support-Vector Regression

1. Introduction

1.1. Motivation

The increasing competitiveness on global markets demands for higher quality products at lower prices. To meet these high expectations of the market, an accurate estimation of the final product price throughout all stages of its development cycle is crucial. Roughly 75 % of the costs are fixed after the design phase, which makes this the most important time for cost estimation, despite the uncertainties at this point of the development. [72]

The complexity of this task raises further when the products are manufactured by suppliers. Successful negotiations with the suppliers of such purchased products can make the difference between a profitable project or an unprofitable one for any company. Overpriced products lead to an unnecessary profit loss and therefore to a decreasing competitiveness of the company. Hence, knowing the costs of the products in advance can be extremely valuable for every company to get an additional edge over the competitors on the market. Nevertheless, the precise prediction of the final cost is difficult even without the additional uncertainties that occur by the manufacturing by a supplier.

In this scenario, most traditional cost estimation approaches have their difficulties to predict the cost with an acceptable precision. Analytical techniques require detailed information about the manufacturing process, like the machine and labour costs, which are usually only available to the supplier. Parametric and non-parametric estimation models, on the other hand, are negatively affected by non-product related cost drivers in the historical data, used to generate them. Prices for purchased products are influenced by additional factors like the profit margin of the supplier, its free capacities, available machinery and many more. All of these add additional noise to the historical data and therefore make an accurate prediction more complicated.

1.2. Research Goals

The goal of this thesis is the design of a new approach to accurately estimate the cost of purchased products. This approach shall overcome the previously mentioned difficulties with classical cost estimation methods for products manufactured by a supplier. For this purpose, a hybrid cost estimation approach using Machine Learning (ML) on historical data, which was cleaned beforehand using different outlier removal methods based on other cost estimation techniques, is proposed. Herein, the main focus lies on the development of these new outlier removal methods for this specific product cost estimation scenario and the evaluation of their influence on the ML model and the final cost prediction. To achieve this, a case study on deep-drawn sheet-metal parts is performed and used to gather insights in the behaviour of the outlier removal methods and the new approach as a whole. Detailed analysis of the performance of different ML models to identify the most suiting one for product cost estimation is beyond the scope of this thesis and can be found elsewhere in literature (ref. Section 3.1).

1.3. Structure of Thesis

After the introduction of this thesis, Chapter 2 covers the theoretical background of the concepts used in the proposed approach. The fundamentals of the ML models utilized for the cost estimation and the techniques applied in the outlier removal process are covered and explained. Afterwards, an overview on the different product cost estimation and outlier removal methods used in literature and their individual strengths and weaknesses is given in Chapter 3. The details of the proposed hybrid cost estimation approach are described in Chapter 4. First, the general approach is introduced, followed by a more detailed presentation of the individual elements, especially the different outlier removal methods. This is followed by the concrete implementation of the approach and its evaluation on a case study covered in Chapter 5. Finally, in Chapter 6 the inferred conclusions of the new cost estimation approach are given along with future research questions.

2. Background

This chapter covers the fundamentals of the concepts applied in this thesis. It can be used as reference for details of the algorithms, models and methods used in the proposed cost estimation approach.

2.1. Support-Vector Regression

A commonly used model for classification and regression tasks is the Support-Vector Machine (SVM). The basic idea of SVMs is the separation (classification) or fit (regression) of data points with a hyperplane. This idea was first introduced by Vapnik and Chervonenkis [70] for binary classification problems. Later it got extended to regression tasks [25]. Reasons for their popularity are, inter alia, a well researched theoretical background, their formulation, which guarantees to find the global optimum, and their versatility, which is achieved by the numerous kernel functions that can be used to solve non-linear problems.

Mathematically the Support-Vector Regression (SVR) can be expressed as a minimization problem on a set X of size N containing feature vectors $\boldsymbol{x}_i \in R^n$ and their corresponding real-valued outputs $y_i \in R$:

$$\begin{aligned}
\underset{\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i + C\sum_{i=1}^{N}\xi_i^* \\
\text{subject to} \quad & \boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b - y_i \leq \epsilon + \xi_i, \\
& \boldsymbol{w}^T\phi(\boldsymbol{x}_i) - b + y_i \leq \epsilon + \xi_i^*, \\
& \xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, N,
\end{aligned} \tag{2.1}$$

with the weight vector $\boldsymbol{w}$, the constant b, the regularization parameter $C > 0$ and $\epsilon > 0$ [12][69, Chapter 11]. ϵ defines an interval $[y_i - \epsilon, y_i + \epsilon]$ in which a deviation between the predicted and actual value is not penalised — hence, the name ϵ-SVR. ξ_i and ξ_i^* are called slack variables and are needed to allow for a non-perfect fit, whereas $\phi(\boldsymbol{x_i})$ is the mapping of $\boldsymbol{x}_i$ in another space, defined by the kernel function used for the SVR.

The problem from (2.1) can also be formulated in its equivalent dual form:

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\alpha^*}}{\text{minimize}} \quad & \frac{1}{2}\|(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\|^2 K(\boldsymbol{x}_i,\boldsymbol{x}_j) + \epsilon\sum_{i=1}^{N}(\alpha_i+\alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i+\alpha_i^*) \\ \text{subject to} \quad & \sum_{i=1}^{N}(\alpha_i-\alpha_i^*) = 0, \\ & 0 \le \alpha_i, \alpha_i^* \le C, \quad i = 1,\dots,N\,, \end{aligned} \tag{2.2}$$

with the kernel function $K(\boldsymbol{x}_i,\boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ [12][69, Chapter 11].

In case of a linear SVR, the kernel function is calculated by the dot product $K(\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j$, which means a hyperplane is fitted to the original data points without any transformation. For non-linear fits, the original data is transformed into another space in which the data can be fitted with a hyperplane. For this transformation, a variety of different kernels like the polynomial $K(\boldsymbol{x}_i,\boldsymbol{x}_j) = (\boldsymbol{x}_i^T\boldsymbol{x}_j)^d$ or the Radial Basis Function (RBF) $K(\boldsymbol{x}_i,\boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2\right)$ can be used. In these cases, the hyperplane is fitted to the transformations of the original feature vectors resulting in a non-linear fit in the original space. One advantage of SVMs is that the calculation of the mapping of the feature vectors $\phi(\boldsymbol{x}_i)$ is not necessary; only $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ needs to be computed. Depending on the complexity of the mapping, this can speed up the computation tremendously, e.g. for the RBF kernel where the feature vectors are mapped into an infinite feature space.

2.2. AdaBoost Forest

Boosting is a technique which combines weak estimators to an ensemble with a performance superior to any of the individual models. This is achieved by using different subsets of the training data for each estimator. In comparison to bagging where these subsets are drawn randomly, boosting focuses more on the data samples which are hard to predict. This focus is realized by training multiple estimators iteratively, while the probability of a data sample being chosen for the training of the current estimator depends on the prediction error for this sample of the previous estimator. The first algorithm using this idea was proposed by Schapire in 1990 [63]. Several other implementations of this idea followed until the most commonly used *AdaBoost* algorithm was introduced by Freund and Schapire in 1996 [30].

In this thesis, the *AdaBoost.R2* algorithm introduced by Drucker [24] with a slight extension is used. Given a set X of size N containing multiple feature vectors $\boldsymbol{x}_i \in R^n$ and their corresponding real-valued outputs $y_i \in R$, this algorithm initially assigns to each sample a weight $w_i = 1$. Then the following procedure is repeated until the desired number of estimators M is reached:

Algorithm 1: Training procedure according to AdaBoost.R2

```
**for** $m = 1$ **to** $M$ **do**
    **for** $\boldsymbol{x_i} \in X$ **do**
        $p_i \longleftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$
    draw $N$ samples from $X$ according to their probability $p_i$
    train estimator on drawn samples
    **for** $\boldsymbol{x_i} \in X$ **do**
        $L_i^{(m)} \longleftarrow \frac{|\hat{y}_i^{(m)} - y_i|}{D}$
    $\bar{L}_m = \sum_{i=1}^{N} L_i^{(m)} p_i$
    $\beta_m = \bar{L}_m / 1 - \bar{L}_m$
    **for** $\boldsymbol{x_i} \in X$ **do**
        $w_i \longleftarrow w_i \beta_m^{\alpha(1-L_i)}$
```

For the training of a new estimator m, N samples are drawn with a probability

$$p_i = \frac{w_i}{\sum_{i=1}^{N} w_i} \tag{2.3}$$

from X with replacement. After training of the estimator on this subset, the loss $L_i^{(m)}$ for each sample is calculated with

$$L_i^{(m)} = \frac{|\hat{y}_i^{(m)} - y_i|}{D}, \tag{2.4}$$

where $\hat{y}_i^{(m)}$ is the prediction of estimator m for a particular sample $\boldsymbol{x}_i$, y_i is its real target value and $D = \sup |\hat{y}_i^{(m)} - y_i|,\ i = 1, \ldots, N$. Subsequently, the average loss

$$\bar{L}_m = \sum_{i=1}^{N} L_i^{(m)} p_i \tag{2.5}$$

and the uncertainty measure

$$\beta_m = \frac{\bar{L}_m}{1 - \bar{L}_m} \tag{2.6}$$

are determined. With β_m the weight of each sample for the next round can be adjusted by the following rule: $w_i \to w_i \beta_m^{\alpha(1-L_i)}$. The shrinkage parameter or learning rate $\alpha \in (0, 1]$ is added here to the original algorithm by Drucker as an additional measure against overfitting. Having M estimators after the training, the final prediction is determined from the individual predictions of each estimator $\hat{y}_i^{(m)}$ by the weighted median defined by

$$\hat{y}_i^{(\text{ensemble})} = \inf \left\{ \hat{y}_i \in \hat{Y}_i : \sum_{m: \hat{y}_i^{(m)} \leq \hat{y}_i} \log\left(\frac{1}{\beta_m}\right) \geq \frac{1}{2} \sum_{m=1}^{M} \log\left(\frac{1}{\beta_m}\right) \right\}. \tag{2.7}$$

In theory, the AdaBoost algorithm can be applied to any estimator. However, a common choice is a Decision Tree regressor, since it is computationally cheap and because of its simple hyperparameter tuning. Using a Decision Tree for regression means using a sequence of binary splits on features to partition the input space of a data set X [7, Chapter 8]. Figure 2.1 shows a graphical representation of this process: Node n_1 contains the whole set X which gets split into two subsets in n_2 and n_3. The splitting is then repeated with the child nodes until the resulting nodes meet certain stopping criteria. The end node of each branch is called terminal or leaf node, represented as a quadratic node in the figure. These leaf nodes contain one or multiple elements of X.

Given a node n with a subset S of X, any possible split $\tau(f, t)$ by feature f with threshold t divides S into the two subsets

$$S_L = \{(\boldsymbol{x}_i, y_i) \mid x_f \leq t\} \tag{2.8}$$

and

$$S_R = \{(\boldsymbol{x}_i, y_i) \mid x_f > t\}. \tag{2.9}$$

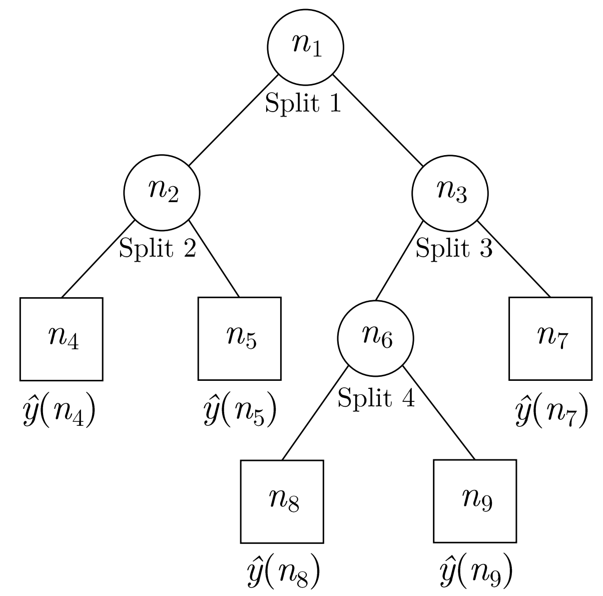


Figure 2.1.: Structure of a Decision Tree Regressor [7, Chapter 8]

The best split $\tau^*(f, t)$ maximizes

$$\Delta R(S, \tau) = R(n) - \frac{|S_L|}{|S|} R(n_L) - \frac{|S_R|}{|S|} R(n_R)\,, \tag{2.10}$$

where $R(n)$ is the Mean Squared Error (MSE) of node n

$$R(n) = \frac{1}{|S|} \sum_{i \in S} (y_i - \hat{y}(n))^2 \tag{2.11}$$

and $\hat{y}(n)$ is the predicted target value of node n calculated by the mean of all y_i in this node. [7, Chapter 8][56]

Splitting according to the best split is performed iteratively until a certain stopping criterion is met. The following criteria are commonly used:

- Maximum allowed depth in tree is reached.
- Maximum number of allowed leaf nodes is reached.
- Any further split would result in a node with less samples than allowed.
- Any further split would result in $\Delta R(S, \tau)$ being below a certain threshold.

2.3. Multilayer Perceptron

The research on Artificial Neural Networks (ANNs) began in the early 1940's and was heavily inspired by the human nervous system. McCulloch and Pitts [50] showed in 1943 that any logical expression can be described by a network of neurons. Since then, a lot of researchers contributed to develop ANNs for all kind of different tasks.

For regression, a special type of feed-forward network, called Multilayer Perceptron (MLP), is commonly used. MLPs are an expansion of the basic Perceptron introduced by Rosenblatt [58] in 1958. Figure 2.2 shows the basic structure of a MLP.

The network structure of MLPs can be described with graphs. Kruse et al. [42, Chapters 4–5] define a r-layered Perceptron as a directed graph $G = (U, C)$ with vertices $u \in U$ called units or neurons and edges $c \in C$ called connections. The vertices either belong to the set of input neurons U_{in}, the set of output neurons U_{out} or the set of hidden neurons U_{hidden}. For these sets the following conditions hold true:

$$U = U_{\text{in}} \cup U_{\text{out}} \cup U_{\text{hidden}} \,, \tag{2.12}$$

$$U_{\text{in}} \neq \emptyset \,,\; U_{\text{out}} \neq \emptyset \,,\; U_{\text{in}} \cap U_{\text{out}} = \emptyset \,,\; U_{\text{hidden}} \cap (U_{\text{in}} \cup U_{\text{out}}) = \emptyset \,, \tag{2.13}$$

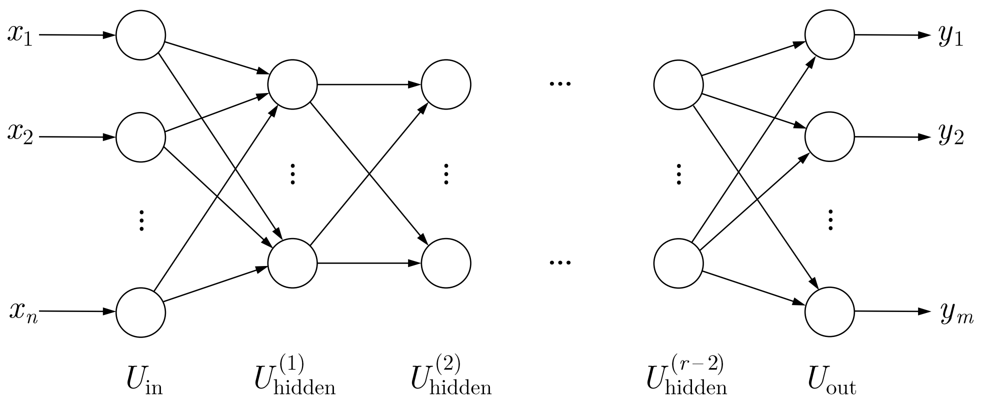
$$U_{\text{hidden}} = U_{\text{hidden}}^{(1)} \cup \ldots \cup U_{\text{hidden}}^{(r-2)} \,, \tag{2.14}$$

$$\forall 1 \leq i < j \leq r-2 : \quad U_{\text{hidden}}^{(i)} \cap U_{\text{hidden}}^{(j)} = \emptyset \,, \tag{2.15}$$

$$C \subseteq \left(U_{\text{in}} \times U_{\text{hidden}}^{(1)}\right) \cup \left(\bigcup_{i=1}^{(r-3)} U_{\text{hidden}}^{(i)} \times U_{\text{hidden}}^{(i+1)} \right) \cup \left(U_{\text{hidden}}^{(r-2)} \times U_{\text{out}}\right) \,. \tag{2.16}$$

Each neuron can only be part of one layer and the input and output layer need to contain at least one neuron each. The hidden part of the network consists of one or multiple single layers containing one or multiple neurons each. Connections are only possible between consecutive layers, which means that information is transferred from the input layer to the output layer without any cycles. This classifies MLPs as feed forward networks.

Furthermore, to each connection $(u, v) \in C$ from neuron u to neuron v a weight w_{vu} is assigned and each neuron $u \in U$ possesses three additional values computed by three functions inside the neuron:



x_1
x_2
x_n
y_1
y_2
y_m
U_{in}
$U^{(1)}_{\text{hidden}}$
$U^{(2)}_{\text{hidden}}$
$U^{(r-2)}_{\text{hidden}}$
U_{out}

Figure 2.2.: Structure of a r-layered Perceptron [42, Chapter 5]

- network input net_u, network input function $f^{(u)}_{\text{net}}$
- activation act_u, activation function $f^{(u)}_{\text{act}}$
- output out_u, output function $f^{(u)}_{\text{out}}$

The network input function processes the incoming inputs at a neuron. For neurons in the hidden and output layer it is the weighted sum of all inputs minus a bias term θ:

$$\forall u \in U_{\text{hidden}} \cup U_{\text{out}} : \quad f^{(u)}_{\text{net}} = \boldsymbol{w}_u^T \mathbf{in}_u - \theta_u = \sum_{p \in \text{pred}(u)} (w_{up} \text{out}_p) - \theta_u \,, \quad (2.17)$$

where $\mathbf{in}_u$ is the input vector, $\boldsymbol{w}_u$ is the associated weight vector for all incoming connections for neuron u and θ_u is a constant called bias. With the definition that all neurons of the previous layer having a connection to neuron u are called predecessors of u, $\text{pred}(u) = \{p \in U \mid (p,u) \in C\}$, the network input function can be reformulated as the weighted input of all outputs of the predecessor neurons subtracted by the bias. For neurons of the input layer, the network input is one dimension of the input vector $\boldsymbol{x}$:

$$\forall u \in U_{\text{in}} : \quad f^{(u)}_{\text{net}} = x_i \,. \quad (2.18)$$

The activation function uses the network input to calculated how much the neuron gets activated. Various different functions are used for this purpose. Some of the most common ones are shown in Appendix A Figure A.1. Usually, activation functions are monotonic and differentiable, but these criteria are not mandatory. The output function of a neuron is used for scaling of the activation. Therefore, mostly linear functions are used. [42, Chapters 4–5]

Assuming a set X of size N containing multiple feature vectors $\boldsymbol{x}_i \in R^n$ and their corresponding real-valued outputs $y_i \in R$, the input layer of the MLP consists of n neurons. Since the target variable is one-dimensional, the output layer of the MLP contains only one neuron. The hidden layers in between can vary in their number and size according to the complexity of the problem.

Given this set X, for the training of a MLP the weights of each connection and the bias of each neuron are first initialised — often randomly — and then iteratively adjusted to reduce the error between the prediction $\hat{y}_i$ of the network and the target variable y_i. This is achieved by minimizing a loss function l which captures the prediction error. The most wide-spread loss function is the MSE calculated by

$$\text{MSE} = \frac{\sum_{i=1}^{N} \left(\hat{y}_i - y_i\right)^2}{N} . \tag{2.19}$$

Besides the MSE, a variety of other loss functions can be used. Nevertheless, the MSE has some properties that make it so versatile; e.g., it is fully differentiable in contrast to other losses, like the Mean Absolute Error. This simplifies the computations in the training phase of the network. Further, it stronger penalises larger errors due to the quadratic error term, which helps the network to converge faster.

During the training the adjustments of the weights are calculated by a variation of gradient descent and the so-called error backpropagation. To simplify the calculations, the biases are usually converted into an additional constant input of one with a corresponding weight for each neuron. The gradient descent method computes the gradient of the loss function w.r.t. the current weights in the network and updates them with a small step in the opposite direction of the gradient (since the loss should be minimized and the gradient points in the direction of the steepest increase). The gradient of the loss function l for neuron u can be computed with

$$\nabla_{\boldsymbol{w}_u} l = \frac{\partial l}{\partial \boldsymbol{w}_u} = \frac{\partial l}{\partial \text{out}_u} \frac{\partial \text{out}_u}{\partial \text{act}_u} \frac{\partial \text{act}_u}{\partial \text{net}_u} \frac{\partial \text{net}_u}{\partial \boldsymbol{w}_u} , \tag{2.20}$$

where $\frac{\partial \text{net}_u}{\partial \boldsymbol{w}_u}$ equals the inputs of neuron u. For neurons in the output layer, the other terms are only dependent on the functions used for output, activation and loss and hence, can be calculated directly. For neurons in the hidden layers, the output influences the loss indirectly through the successor neurons of u, that are $\mathrm{succ}(u) = \{s \in U \mid (u, s) \in C\}$. Hence, $\frac{\partial l}{\partial \mathrm{out}_u}$ depends on all the neurons that are between neuron u and the output layer:

$$\frac{\partial l}{\partial \mathrm{out}_u} = \left(\sum_{s \in \mathrm{succ}(u)} \frac{\partial l}{\partial \mathrm{out}_s} \frac{\partial \mathrm{out}_s}{\partial \mathrm{act}_s} \frac{\partial \mathrm{act}_s}{\partial \mathrm{net}_s} \frac{\partial \mathrm{net}_s}{\partial \boldsymbol{w}_s} w_{su} \right) . \tag{2.21}$$

With this formula, the gradient can be computed layer-wise from the output to the input layer, which is why this method is called backpropagation. After all gradients are computed, the weights of each neuron are updated according to

$$\Delta \boldsymbol{w}_u = -\eta \nabla_{\boldsymbol{w}_u} l \, , \tag{2.22}$$

with the real-valued learning rate $\eta \in (0, 1]$. [42, Chapter 5]

The update of the weights can be performed after each training sample (online learning), after one iteration over the whole training data (batch learning) or after multiple samples (mini-batch learning). For batch and mini-batch learning the individual updates are accumulated to perform less frequent updates.

One of the reasons for the popularity of MLPs is their flexibility. Given enough hidden neurons, any function can be fitted, regardless of its complexity. However, this characteristic is also a weakness of MLPs as it can result in overfitting. A common technique to regularize ANNs, besides reducing the number of hidden neurons, is using Dropout. The idea behind Dropout is to randomly drop neurons and their connections from the network during training and was proposed by Hinton et al. [33]. For each training step, every hidden neuron is present with the probability p. Non-present neurons and their connections are deleted for this step. The Dropout is only active while the network is trained; after training, all neurons are active but all weights are adjusted by the factor p to compensate the additional neurons. This method can be interpreted as a computationally efficient way to train many different MLPs and using their averaged predictions in application. Instead of describing the probability p of a neuron to be present, often the Dropout rate $1 - p$ is used to indicate how likely it is for a neuron to be dropped.

2.4. Data Envelopment Analysis

In the field of operations research the Data Envelopment Analysis (DEA) is often used to compare entities, like hospitals, educational institutions or companies, based on their efficiency. It was initially proposed by Charnes, Cooper and Rhodes in 1978 [13]. The interesting characteristic of the DEA is its possibility to take multiple inputs and outputs for each entity, also called Decision Making Unit (DMU), into account. The efficiency is defined as the ratio of output to input, which makes the DEA approach extremely flexible in its application, since any numerical measure can be used as an input or output. This was also shown by Defersha, Salam and Bhuiyan [22] who proposed a product cost estimation approach based on a DEA.

Given a data set X of N DMUs, with an input vector $\boldsymbol{x}_i \in R^n$ and an output vector $\boldsymbol{y}_i \in R^m$, the original DEA model, called *CCR* model, maximizes the efficiency, which is the ratio of the weighted output to the weighted input with the two weight vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, for every DMU d according to

$$\begin{aligned} \underset{\boldsymbol{u},\boldsymbol{v}}{\text{maximize}} \quad & \theta = \frac{\boldsymbol{u}_d^T \boldsymbol{y}_i^{(d)}}{\boldsymbol{v}_d^T \boldsymbol{x}_i^{(d)}} \\ \text{subject to} \quad & \frac{\boldsymbol{u}_d^T \boldsymbol{y}_i^{(k)}}{\boldsymbol{v}_d^T \boldsymbol{x}_i^{(k)}} \leq 1 \,, \quad k = 1, \dots, N \,, \\ & u_1^{(d)}, \dots, u_m^{(d)} \geq 0 \,, \\ & v_1^{(d)}, \dots, v_n^{(d)} \geq 0 \,. \end{aligned} \tag{2.23}$$

The constraints imply that the efficiency value θ lies in the interval $[0, 1]$, assuming strongly positive inputs and outputs. In addition, they ensure that the best weights for DMU d would not lead to an efficiency outside of $[0, 1]$, when they are applied for any other DMU k in the data set. [18, Chapter 2]

The DEA formulation in (2.23) can be reformulated by The equivalent linear formulation

$$\begin{aligned} \underset{\boldsymbol{u},\boldsymbol{v}}{\text{maximize}} \quad & \theta = \boldsymbol{u}_d^T \boldsymbol{y}_i^{(d)} \\ \text{subject to} \quad & \boldsymbol{v}_d^T \boldsymbol{x}_i^{(d)} = 1 \,, \\ & \boldsymbol{u}_d^T \boldsymbol{y}_i^{(k)} \leq \boldsymbol{v}_d^T \boldsymbol{x}_i^{(k)} \,, \quad k = 1, \dots, N \,, \\ & u_1^{(d)}, \dots, u_m^{(d)} \geq 0 \,, \\ & v_1^{(d)}, \dots, v_m^{(d)} \geq 0 \,. \end{aligned} \tag{2.24}$$

In this representation the constraint of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ being strongly positive can be relaxed to a semipositive constraint, allowing for zero values in the vectors unless all components are zero. [18, Chapter 2]

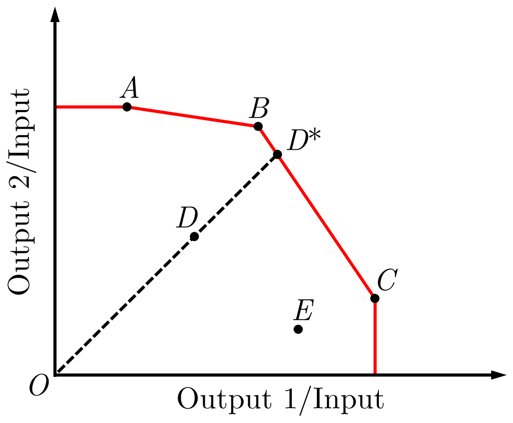


Figure 2.3.: DEA example with two outputs and one input [18, Chapter 1]

Figure 2.3 shows a simple example with two outputs and one input. The DMUs A, B and C are efficient ($\theta = 1$) and build an efficiency frontier. Any point on this frontier is a semipositive combination of these three points and considered efficient. Since all DMUs are compared to the efficiency frontier, all points within the area enclosed by the efficiency frontier and the axes — in this example the DMUs D and E — are considered inefficient ($\theta < 1$). The efficiency value of D can be computed with

$$\theta_D = \frac{\overline{OD}}{\overline{OD^*}}\,, \tag{2.25}$$

where D^* is the radial projection of D onto the efficiency frontier [18, Chapter 1].

One downside of this original definition is its assumption of constant returns to scale, meaning that for every DMU with input $\boldsymbol{x}_i$ and output $\boldsymbol{y}_i$ a DMU with $c\boldsymbol{x}_i$ and $c\boldsymbol{y}_i$ for any positive c is possible. However, such behaviour is not always observed in reality; manufacturing processes or even companies usually only scale with constant returns for minor changes in size. To make the DEA method also applicable to problems with variable returns to scale, the *BCC* model was developed by Banker, Charnes and Cooper [5]. This model modifies the DEA so that the efficiency frontier becomes a convex hull and any efficient point has to be a convex combination of the efficient DMUs. The linear formulation of the BCC model adds one scalar u_0 to the CCR model (2.24) to obtain

$$\begin{aligned}
\underset{\boldsymbol{u},\boldsymbol{v}}{\text{maximize}} \quad & \theta_{\text{BBC}} = \boldsymbol{u}_d^T \boldsymbol{y}_i^{(d)} - u_0 \\
\text{subject to} \quad & \boldsymbol{v}_d^T \boldsymbol{x}_i^{(d)} = 1\,, \\
& \boldsymbol{u}_d^T \boldsymbol{y}_i^{(k)} - u_0 \leq \boldsymbol{v}_d^T \boldsymbol{x}_i^{(k)}\,, \quad k = 1, \dots, N\,, \\
& u_1^{(d)}, \dots, u_m^{(d)} \geq 0\,, \\
& v_1^{(d)}, \dots, v_m^{(d)} \geq 0\,,
\end{aligned} \tag{2.26}$$

where u_0 is free of any constraints. This change incorporates the fact that the achievable maximum efficiency might change when the inputs are scaled up or down. [18, Chapter 4]

2.5. Kolmogorov-Smirnov Test

To test whether a data set follows a specific distribution, there are multiple statistical tests available. One of the most common ones for non-normal data is the two-sided Kolmogorov-Smirnov (KS) test. As the name suggests, this test was developed by Kolmogorov [41] and Smirnov [66]. It is a non-parametric test which computes a distance between the Empirical Distribution Function (EDF) $F_N(x)$ and the Cumulative Distribution Function (CDF) $F(x)$ of a specific continuous distribution. Given a set X of size N of data points, the EDF is described by

$$F_N(x) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{(-\infty,x]}(X)\,, \tag{2.27}$$

with

$$\mathbb{1}_{(-\infty,x]}(X) = \begin{cases} 1 & \text{if} \quad X \leq x \\ 0 & \text{otherwise}\,. \end{cases} \tag{2.28}$$

The KS statistic is the maximum distance D_N between the CDF of the tested distribution and the EDF, as seen in Figure 2.4. Thus, it can be calculated by [31, Chapter 4]

$$D_N = \sup_x |F_N(x) - F(x)|\,. \tag{2.29}$$

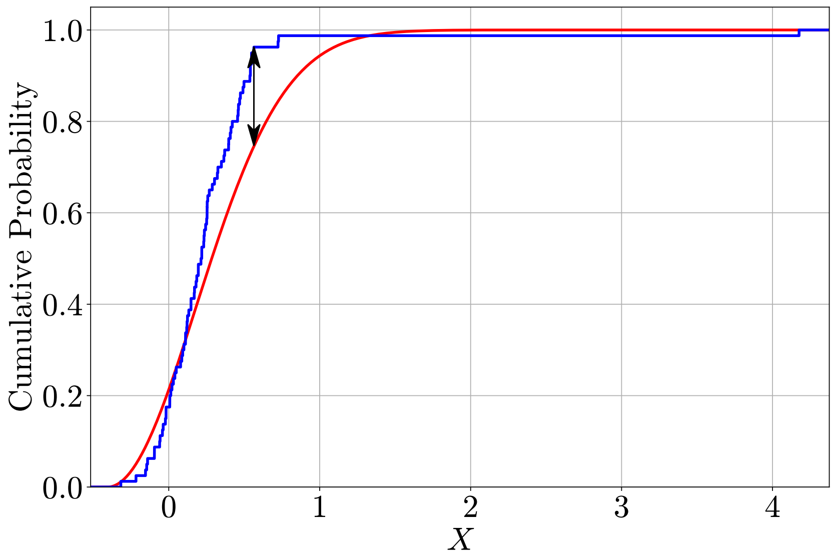


Figure 2.4.: Kolmogorov-Smirnov statistic (*black arrow*) of the empirical distribution function (*blue*) of data from the case study compared to a cumulative distribution function of a fitted Rayleigh distribution (*red*)

The null hypothesis H_0 of the KS test is that all data points are drawn from the tested CDF:

$$H_0: \quad \forall x: \ F_N(x) = F(x)\,. \tag{2.30}$$

Consequently, the alternative hypothesis H_1 of the two-sided test is

$$H_1: \quad \exists x: \ F_N(x) \neq F(x)\,. \tag{2.31}$$

One major advantage of this test is that $\sqrt{N}D_N$ converges to the Kolmogorov distribution for $N \to \infty$ if the null hypothesis is true and the tested distribution continuous. Consequently, the test statistic D_N is independent of the tested CDF and the p-value to test the null hypothesis is defined as

$$P(D < K \mid H_0) = 2\sum_{i=1}^{\infty} (-1)^{i-1} \exp\left(-2i^2D^2\right), \tag{2.32}$$

where $D = \sqrt{N}D_N$ and K is a random variable following the Kolmogorov distribution. [31, Chapter 4]

A disadvantage of the KS test is its tendency to overestimate the p-values when the distribution parameters are estimated from the data set itself. This means the formula for the p-value in (2.32) is not valid for CDFs which have been fitted to the data set beforehand and it has to be calculated with Monte Carlo methods [21, Chapter 4]. The method applied in this thesis was proposed by Clauset, Shalizi and Newman in 2009 [16] and is described in Algorithm 2.

Algorithm 2: Simulation procedure for a corrected estimate of the p-value for the Kolmogorov-Smirnov test

```
estimate the distribution parameters from $N$ data samples
calculate KS statistic $D_N$ for the estimated distribution
counter $\longleftarrow 0$
for $i = 1$ to $M$ do
    sample data set of size $N$ from the original estimated distribution
    estimate the distribution parameters for the sampled data set
    calculate KS statistic $D_N^{(i)}$
    if $D_N^{(i)} > D_N$ then
        counter $\longleftarrow$ counter $+ 1$
$P(H_0) \longleftarrow \frac{\text{counter}}{M}$
```

3. State of the Art

The following sections present methods and techniques used for product cost estimation and outlier detection. To give an overview on the methods in these fields, the most common and recent approaches and their individual strengths and weaknesses are presented.

3.1. Product Cost Estimation

Product cost estimation is of great importance for any producing company. It is essential for a competitive and profitable business. Because of this, large efforts have been put into the research of new and more accurate estimation approaches. According to Niazi et al. [53], product cost estimation techniques can be classified into two main categories: quantitative and qualitative methods.

Quantitative cost estimation techniques describe the product cost as a function of cost-related variables. They can be further divided into parametric, non-parametric and analytical approaches [38]. In parametric models, the type of relationship/function between the cost drivers and the cost is defined a priori. According to this predefined function, the model has a fixed number of parameters. In contrast, non-parametric approaches use models which fit an unknown function to the data. This allows for more flexibility in the description of the relationship between the cost drivers and the cost. In these models the number of parameters is not fixed and can vary depending on the data set.

Parametric and non-parametric models are built with statistical or ML techniques and the use of historical data. Their cost drivers usually contain variables which are directly linked to product features (e.g. weight, material). Recent research revolves around developing parametric cost models for different product types. Martinelli et al. [49] proposed a parametric approach for gas turbines, while Cavalieri, Maccarrone and Pinto [11] developed one for brake disks, to name two examples. With the rise of ML, in particular ANNs, these techniques were investigated for cost estimation intensively. Ning et al. [54] used deep-learning techniques to automatically extract product features, which were then used to train different ML models. In addition, ML-based cost prediction was used for a variety of different product types [26, 44, 46]. But also methods from other fields were applied, like the DEA approach for predicting the lowest achievable cost for landing gears of aircraft proposed by Defersha, Salam and Bhuiyan [22].

In contrast to the data-driven approach of parametric and non-parametric techniques, analytical models interpreted the cost as a sum of resources and operations [15, 53]. They break up the product cost in partial costs for material, manufacturing and additional activities in varying degrees of detail, depending on the focus of the particular model. This also includes cost drivers which are not directly linked to the product, e.g. logistic or detailed machinery cost. Each of the partial costs are described by its own, either with an individual function or other methods like look-up tables etc., and then summed up to obtain the total product cost. This bottom-up approach allows for a fine differentiation between manufacturing scenarios, contrary to parametric or non-parametric models, where the non-product-related cost drivers have to be incorporated in the product features. Recent literature consists of several applications of the analytical cost method to various product types [29, 47].

Qualitative product cost estimation relies on experience and historical cost data. There are two types of qualitative approaches: intuitive and analogical techniques [15, 53]. Intuitive cost estimation uses expert knowledge and data from previous products. Based on similar existing products, the product cost is intuitively estimated by comparisons performed by experts.

Analogical techniques also infer their cost estimation from a comparison with similar historical products but with the help of decision-support-systems. These systems use rules, constraints and product databases to help adjusting the cost based on the difference to the historical products [15]. Recent analogy-based approaches rely on Case-Based Reasoning, Decision Trees, Genetic Algorithms or probability distribution optimization methods [4]. Further research was done on new technologies to identify similar products. Ćwikła and Bańczyk [20] proposed a semantic net approach to search for similar products in a database and Mrozinski et al. [52] used a new technique based on k-Nearest Neighbours.

Technique	Product Knowledge	Expert Knowledge	Historical Data
Parametric	Medium	High	Medium
Non-parametric	Medium	Low	High
Analytical	High	High	Low
Intuitive	Low	High	Medium
Analogical	Low	Medium	High

Table 3.1.: Requirements for product cost estimation techniques

Table 3.1 shows a rough overview of the requirements for the different product cost estimation approaches. In general, it can be recorded that qualitative estimation techniques are more applicable in earlier design phases of the product, where the design is not yet finalised. They compensate their lack of knowledge about the final design with expert knowledge or big data sets of historical data. Quantitative methods usually require a more mature design of the product. The three types differ mainly in the need of expert knowledge and historical data.

To combat the disadvantages of each product cost estimation technique, suitable ways to combine them were investigated. Such hybrid approaches try to lower the different requirements and/or combine strengths of the individual techniques. An example for such a hybrid, was developed by Díaz et al. [23], who used a method based on analogical and parametric techniques. Their method uses clustering to identify similar groups of products and then performs a polynomial regression on each cluster. Another example is the combination of parametric and non-parametric techniques, which Sajadfar and Ma [62] applied to estimate the cost of welded products.

3.2. Outlier Detection

Like any other field dealing with data, product cost estimation struggles with outliers in the cost data. Since outliers are such a common issue, numerous methods have been developed for their identification. The precise definition of outliers can vary based on the field and the specific task. Generally, they can be described as data points not aligning with the expected behaviour of the data set, having values that are far off the expected or average value or showing dissimilarity to the normal data in some characteristics [67, Chapter1]. Since outliers are prominent in almost any data analysis, over the time, a substantial amount of research was carried out on their detection. Each scientific community came up with their own ways and methods to deal with these data points, but overall they can be classified in three categories [2, Chapter 1][6]:

- Supervised methods assume at least partially labeled data for normal data and outliers.
- Semi-supervised methods work an data sets where the non-outliers are (partially) labeled.
- Unsupervised methods try to distinguish outliers without any labeled data.

Further, only the unsupervised methods are investigated, as for regression tasks this is the most common scenario. Aggarwal [2, Chapters 2–6] gives an extensive overview on them, from which the most common ones are briefly summarized in the following paragraphs. Generally, unsupervised outlier detection techniques can be categorized in removal of extreme values, model-based and proximity-based outlier detection.

Techniques which identify extreme values base on the idea that values in a data set are not uniformly distributed and the occurrence of extreme values is unlikely. Therefore, data points with such extreme values have a high probability of being outliers. One simple and commonly used method of this kind is the z-score method [19, 59]. The z-score describes the distance between a value and the mean in relation to the standard deviation. Data points with z-scores over a certain threshold are omitted. In case a Gaussian distribution cannot be assumed, the general concept can be applied to other types of distributions. Two major disadvantages of this technique are that a certain distribution has to be assumed and it can only be applied to univariate data. Hence, multivariate data can only be analysed one feature at a time.

To overcome the restriction on dimensionality, this idea can be extended to the multidimensional space, e.g. with the Mahalanobis distance. The distribution of the data is estimated by a multivariate Gaussian distribution and the Mahalanobis distance between a data point and the mean of the distribution can be interpreted as the distance to the mean in relation to the standard deviations. This approach was used by Laurikkala, Juhola and Kentala [43] for example. Hubert, Debruyne and Rousseeuw [34] proposed a variation of this method, which is more robust to outliers in the initial estimation of the distribution. Nevertheless, these methods still assume a certain distribution type and are therefore only applicable if this assumption is valid.

Another technique to remove extreme values are depth-based approaches. Contrary to the statistical methods, these construct a convex hull around the data and remove the edge points of this hull. This process is repeated iteratively until a certain depth is reached. In this way, the most outer data points are removed without the assumption of any distribution. Examples for this type of outlier detection can be found in [36] and [61].

By design the extreme value approaches can only remove outliers located at the edges of the data set. Nevertheless, outliers can also occur in more central positions.

Another type of outlier identification are the model-based approaches, which can detect outliers independently of their location in the data set. They describe the relationships between the variables with models and detect the data points with high deviation from it. The most basic approach is to fit a hyperplane to the data via Linear Regression. In order to deliver good results, the fitted hyperplane should only reflect the behaviour of the inliers. Since Linear Regression is very sensitive to outliers, especially at the edges, extensive research was done to increase the robustness of the linear fit. An overview over common methods can be found in [59] and [60].

For higher dimensional data, Principal Component Analysis (PCA) might be a better option. This method is used for dimensionality reduction by generating new features (principal components) from linear combinations of the original ones. These principal components are uncorrelated and ordered by their variance [37, Chapter 2][65]. As outliers do not follow the correlations of the rest of the data set, they are identifiable after the transformation with a graphical or distance based approach. PCA for outlier detection is a well researched topic and many examples for its application are available [10, 27, 65].

In addition to linear approaches, also SVMs can be used to fit the data. The main advantage of SVMs is that they can model non-linearities when the kernel-trick is used. This makes them more flexible and applicable for a wider range of scenarios. Two examples for outlier detection via SVMs can be found in [48] and [64].

The most flexible model-based approach are auto-encoders, which use ANNs to encode the data. These encodings can be highly non-linear and can be used to detect outliers in data sets. As the encoding compresses the information, the auto-encoder focuses on the main relationships in the data. Since outliers deviate from these relationships, their deviations before and after their en- and decoding are higher than the ones of inliers [9]. Applications of this method can be found in [57] and [68].

The performance of model-based approaches heavily depends on an accurate representation of the actual relationships in the data. Their quality can suffer from a large amount of outliers in the data and wrong model assumptions. Additionally, these methods are only reasonable if correlations between the features exist.

Proximity-based techniques define outliers by their location in relation to other data points. They can be divided in three subcategories. Cluster-based algorithms like proposed in [28] or [35] find groups of similar data points in the data set. Any point outside a cluster or inside a small enough cluster can be considered an outlier.

Distance-based algorithms use the distance of points to their neighbours to identify outliers. Since outliers are usually isolated, their average distance to their neighbours is high in comparison to other data points. This idea was first introduced by Knorr and Ng [40]. Commonly, variations of k-Nearest Neighbours [32] or an ensemble of Decision Trees like Isolation Forests [45] are used to detect isolated data points.

Both, cluster-based and distance-based approaches do not perform well, when the density of data points strongly varies in the data set. Since these techniques apply a fixed threshold to the whole data set (globally), they struggle to capture local anomalies. This weakness led to the proposal of density-based approaches, which compare the density around a data point to the density in their local neighbourhood. The three most popular approaches of this type are OPTICS [3], the related Local Outlier Factor (LOF) [8] and the Local Correlation Integral (LOCI) [55].

Proximity-based techniques are ideal for data sets with multiple clusters. This can be highly beneficial for classification but is usually not relevant for regression tasks. Additionally, the variability of results between different approaches is high. Therefore, special care in the selection of the algorithm used for a particular data set is advised.

Table 3.2 shows a summary of the strengths and weaknesses of the three different categories of outlier detection techniques.

Type	Advantage	Disadvantage
Extreme Values	Simplistic concept	Only coverage of edges of the data set
	Solid probabilistic foundation	Assumption of a certain distribution
Model-based	Coverage of whole data space	Tendency of overfitting
		Assumption of correlated features
Proximity-based	Coverage of whole data space	High variability in results
	Identification of local outliers	Limited applicability for regression

Table 3.2.: Advantages and disadvantages of outlier detection methods

4. Hybrid Cost Estimation

Using historical data for cost estimation with classical parametric or non-parametric methods has one major disadvantage. All data points are considered as equally valid and therefore it is assumed that all of them reflect the cost drivers equally well. However, this assumption is often not met in reality, especially when the parts are not produced in-house but bought from suppliers. Consequently, variations of the cost based on parameters not related to the product have to be expected. Some of these influencing factors can be the capacity and availability of certain machinery and tools at the suppliers, the success of price negotiations, etc. All of these can lead to deviations from the expected price based on the product-related cost drivers. Furthermore, these additional cost factors are normally not known to the buyer and therefore impossible to predict for future products.

To deal with this problem, this thesis investigates different methods to filter the historical data and eliminate data samples, whose price is dominated by non-product related cost drivers. Based on these outlier detection methods, a proposal for a new hybrid cost estimation approach is developed, which focuses on the prediction of a realistically achievable market price for purchased parts.

The proposed method consists of two main elements, as depicted in Figure 4.1: the filtering of the historical cost data and the model building with ML techniques. The identification of outliers is either based on an analytical cost model, a DEA or both and combines ideas from model-based and statistical outlier detection. The analytical cost model and the DEA are used to model the product cost and compare this estimate with the actual cost. Products with a large discrepancy between the two are then removed from the data set. After this removal, the filtered data is used to build a ML model to estimate the cost.

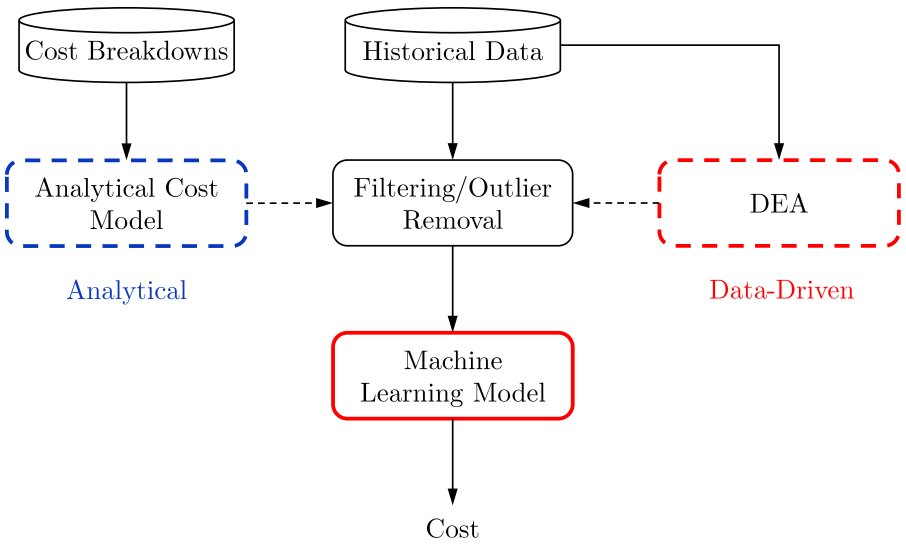


Figure 4.1.: Overview of the proposed cost estimation approach

4.1. Filtering of Historical Data

In this section, the different ways of removing outliers from the historical data set are discussed in detail. Two different ideas are the base for the three proposed outlier removal approaches. The first method uses a simplified analytical cost model to identify outliers, while the second one relies on a DEA. The third approach combines both. For each approach two variants are presented and explained in detail.

4.1.1. Deviation from Analytical Cost Model

One way to remove data points which do not reflect the product-related cost drivers is the comparison of the actual cost (the price the part was purchased for) and the theoretical cost from an analytical cost model. The analytical model breaks down the cost into the individual different cost drivers, like machine costs, labour costs, material costs, etc. The model currently used to capture the cost of the deep-drawn sheet-metal parts in the later introduced case study (ref. Section 5.1) splits the total product cost c_{total} into material costs c_{m}, costs for the manufacturing process c_{p}, packaging costs c_{pkg} and rates for Selling, General and Administrative Expenses r_{SGA} and the profit of the supplier r_{profit}. The formula describing this split is

$$c_{\mathrm{total}} = \frac{c_{\mathrm{m}} + c_{\mathrm{p}}\left(1 + r_{\mathrm{SGA}}\right) + c_{\mathrm{pkg}}}{1 - r_{\mathrm{profit}}}\,. \tag{4.1}$$

The material cost depends on the material needed for the manufacturing process m_{gross}, the price per kilogram for the material p_{m}, how many of the parts are produced with defects and therefore are considered scrap r_{s} and how expensive it is to handle the material r_{h}. In addition, the resale price for the scrap material p_{s} produced has to be considered. The amount of additional scrap material per part is the difference between the gross weight and the weight of the final part m_{net}. With these parameters, the material cost is described by

$$c_{\mathrm{m}} = m_{\mathrm{gross}} \cdot p_{\mathrm{m}}\left(1 + r_{\mathrm{h}}\right)\frac{1}{1 - r_{\mathrm{s}}} - p_{\mathrm{s}}\left(m_{\mathrm{gross}} - m_{\mathrm{net}}\right)\,. \tag{4.2}$$

The process cost is mainly driven by the operating costs of each machine and the labour costs. Important for both of them is the time needed to finish one product, called cycle time t_{c}. Additionally, the hourly costs for the machines c_{M} and the workers c_{w}, the time to set up the machines t_{setUp}, the amount of parts produced in one batch N_{batch} and the time in which the machine is not productive because of delays or maintenance, which is covered in the overall equipment effectiveness OEE, influence the process cost. Given N_{M} different machines, each of them operated by $N_{\mathrm{w}}^{(i)}$ workers, the process cost is given by

$$c_{\mathrm{p}} = \sum_{i=1}^{N_{\mathrm{M}}} c_{\mathrm{MFG}}^{(i)} + \frac{c_{\mathrm{setUp}}^{(i)}}{N_{\mathrm{batch}}} = \sum_{i=1}^{N_{\mathrm{M}}} \left(\frac{t_{\mathrm{c}}^{(i)}\left(c_{\mathrm{M}}^{(i)} + N_{\mathrm{w}}^{(i)} c_{\mathrm{w}}^{(i)}\right)}{\mathrm{OEE}^{(i)}\left(1 - r_{\mathrm{s}}\right)} + \frac{t_{\mathrm{setUp}}^{(i)}\left(c_{\mathrm{M}}^{(i)} + N_{\mathrm{w}}^{(i)} c_{\mathrm{w}}^{(i)}\right)}{N_{\mathrm{batch}}} \right)\,. \tag{4.3}$$

Unfortunately, all the parameters in (4.3) are usually not know for purchased parts. However, they can be estimated by experts with enough know-how in these processes. Nonetheless, since this is a time-consuming and therefore expensive procedure it is not reasonable to estimate the cost breakdowns for all parts in a data set. Instead, reference processes can be identified by a detailed analysis of only a fraction of the data and the corresponding cost breakdowns. These reference processes should reflect typical cost structures for different manufacturing scenarios and enable a simplified calculation of the process cost with

$$c_{\mathrm{p}} = c_{\mathrm{MFG}}^{(\mathrm{ref})} + \frac{c_{\mathrm{setUp}}^{(\mathrm{ref})}}{N_{\mathrm{batch}}} \,. \tag{4.4}$$

The description of the process cost is now based on the manufacturing and set-up costs of the reference process. The batch size remains the only unknown variable. For its estimation, the strongly correlated yearly production volume of the product can be used in combination with the estimated batch sizes of the detailed cost breakdowns. Given these two, the batch size can be computed by linear interpolation of the estimated batch sizes in the cost breakdowns based on the corresponding yearly volume.

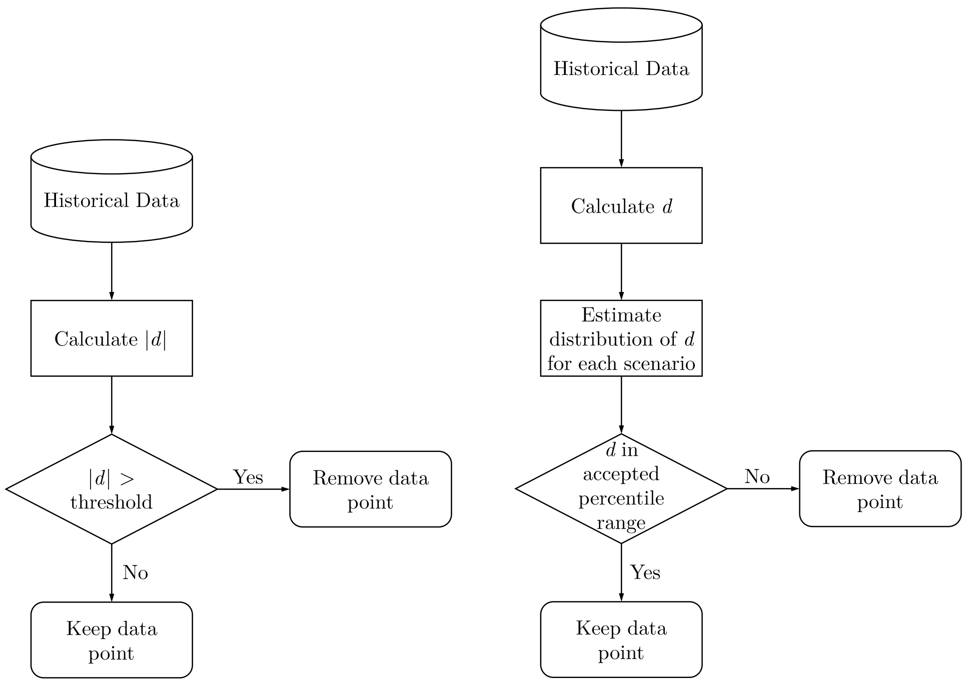
But not only the process cost contains unknown parameters; the process scrap rate r_{s}, the handling rate r_{h} and the gross weight of the material m_{gross}, needed to calculate the material cost in (4.2), are also unknown for historical data. Similar to the cost drivers of the manufacturing process, they can be estimated from the cost breakdowns. For the gross weight, the material yield of the reference processes can be analysed to obtain a ratio between net and gross weight of the parts. The two rates can be extracted directly from the detailed cost analysis. In a similar way, values for the packaging cost and the SGA and profit rates can be extracted. All other cost drivers, like material prices, net weight of the parts and yearly production volumes, are usually known even when the products are not produced in-house.

The final step to complete the analytical cost model is the identification of rules for which parts the reference scenarios can be applied. These rules reflect the physical and economic boundaries of the processes, e.g. the maximum weight of a part machined with a certain equipment or the level of automation still profitable for a given yearly production volume. Given enough detailed cost breakdowns, these rules can be extracted by a thorough analysis or alternatively, they can be given by experts. After finishing these steps, a simplified analytical cost model is built, which only requires inputs usually known for purchased products.

The analytical model can now be used to remove outliers in the historical data to filter out parts for which non-product related cost drivers dominate. A straightforward idea to achieve this goal is to eliminate parts in the data set according to the deviation between the theoretical cost from the analytical model and the actual price payed, as shown in Figure 4.2a. First, the rules defined in the analytical model are applied to identify which reference process is applicable for each part. Then, the theoretical costs of these scenarios are calculated. In case multiple scenarios are valid, an additional rule is needed to decide for only one of them. This rule can be to decide always for the process with the lowest absolute deviation from the historical cost or to favor the cheapest scenario. The former shifts the focus slightly more on the historical data, whereas the latter puts more emphasis on the lowest price achievable according to the analytical cost model. After assigning each part an analytical cost, the relative deviation from this cost

$$d = \frac{c_{\text{historical}} - c_{\text{analytical}}}{c_{\text{analytical}}} \tag{4.5}$$

is calculated and decides whether to keep or to discard the part. Any data with a d-value above a certain threshold will be excluded.



(a) Variant 1 (b) Variant 2

Figure 4.2.: Flowcharts of the two proposed variants for outlier removal based on the deviation to an analytical cost model

The second variant of this analytical approach, shown in Figure 4.2b, tries to incorporate even more theoretical knowledge to filter the data. As mentioned before, the price for purchased products can vary based on non-product related cost drivers. But this spread is not expected to be symmetrical around the analytical cost, since analytical cost models predict idealized costs, which are located at the lower end of the spectrum. Additionally, a lower limit for the deviation should exist, which is located at the point where the supplier starts to operate at a loss. On the other hand, no such limit exists for more expensive prices; they just are more unlikely to occur the more overpriced the parts are (compared to the analytical model). Based on these considerations, the distribution of d (ref. (4.5)) for each reference process is expected to be skewed to the right with a mode near to the analytical model estimation. If now a symmetric cut-off is applied for the lower and the upper end, as in the first approach, most of the outliers identified are more expensive than the analytical model. To overcome this problem, this variant estimates the distribution of the relative deviation from the analytical model and removes a certain percentages of data points from both ends of this distribution. By this method, full control over the percentage of removed data points at each tail is ensured. This percentage can be adjusted based on knowledge about the location of the outliers. After this procedure, the remaining data can be fed to the ML algorithms.

4.1.2. Data Envelopment Analysis Efficiency

Using the DEA as an outlier removal method for cost estimation is inspired by the idea that a good cost estimate should be realistic but also as low as possible. Defersha, Salam and Bhuiyan [22] showed that a DEA approach can be used effectively to predict the product cost from historical data. The approach proposed in this section builds on these findings and develops an outlier removal method based on cost efficiency to achieve a realistic and cost-efficient estimate.

As described in Section 2.4 the DEA maximizes the efficiency or the ratio between weighted output and input parameters for each part under some constraints. This concept can be applied to the cost estimation problem by representing the input as the historical cost of the product c_i and the outputs as the cost drivers/features of this part. These features $\boldsymbol{x}_i$ can be the yearly production volume, the weight, the geometrical complexity, etc. The cost efficiency for part i can then be expressed as

$$\theta_{\text{cost}}^{(i)} = \frac{\boldsymbol{u}_i^T \boldsymbol{x}_i}{v_i c_i} \tag{4.6}$$

or

$$\theta_{\text{cost}}^{(i)} = \frac{\boldsymbol{u}_i^T \boldsymbol{x}_i - u_0}{v_i c_i}\,, \tag{4.7}$$

regarding whether a constant return to scale can be assumed (4.6) or not (4.7). $\boldsymbol{u}_i^T$ is the weight vector for the cost drivers and v_i the weight of the cost. Both are optimized for each part within certain constraints to maximize $\theta_{\text{cost}}^{(i)}$ (ref. Section 2.4). The former equation is called CCR model and implies that the efficiency $\theta_{\text{cost}}^{(i)}$ is not affected by the scaling of input and output. This would presume a part with ten-times the cost drivers would also cost ten-times more. For cases where this assumption cannot be made, the BCC model (4.7) introduces a correcting term u_0. For produced parts, non-linear relationships between cost drivers and cost are prevalent (e.g. the usual non-linear relationship between manufacturing cost and the weight of the product). For these cases the BCC model is probably the better choice, but this has to be decided case by case.

Performing a DEA on historical data assigns each part a value which describes how efficient the cost is reflected in the features of this part. Based on this, the data can be cleaned from parts with low cost efficiency. Since the efficiency is bounded between zero and one, the inefficient parts can be easily removed by defining a minimum cost efficiency. Any part with a value lower than this threshold will be removed.

This approach can be varied to also remove overly efficient parts. By this, the focus on efficiency is lessened and more weight is given to the distribution of the historical data and the fact that some parts might be cheaper than a realistic market price is acknowledged. For this idea, two thresholds for the cost efficiency, a lower and an upper one, are needed and only the data in between is used for the cost estimation. A systematic way to determine the cut-off points is to estimate the distribution of the cost efficiency in the data set and then remove a certain percentage of the data at both tails of the distribution. Depending on how much focus should be laid on the cost efficiency or on the historical data, the percentage of data removed on the upper and lower end of the distribution can be adjusted.

4.1.3. Combination of Data Envelopment Analysis and Analytical Cost Model

The last approach investigated in this thesis is a combination of the two methods presented above. The analytical cost model focuses on a realistic description of the cost for a product, whereas the DEA emphasizes on the most efficient parts in the historical data. Combining these two methods should lead to a cost estimation which reflects both aspects. In order to find a reasonable combination of both approaches, it has to be investigated which data points should be removed. Some parts could be identified as an outlier by their low cost efficiency but not based on their deviation from the analytical cost model. This is possible because the DEA does not take limitations of different manufacturing processes into account and compares all part with each other. Some reference processes can be less cost efficient than others and therefore lead to a worse DEA efficiency. The opposite scenario can also occur. Parts can be decently efficient but vary substantially from their theoretical cost, e.g. because the reference processes identified do not cover all possible manufacturing scenarios and hence, the analytical cost is not an accurate representation. But in both cases the data point should not be removed from the data set, since it still fulfills one of the two objectives. Following this argumentation, parts get only removed when they are flagged as outliers by the DEA and the analytical cost model.

Given the two variants for the analytical and the DEA approach, two variants for their combination are proposed. The first one removes any data from the historical data set that has a greater absolute deviation from the best fitting theoretical cost calculation than a certain threshold and additionally has a lower efficiency than the defined cut-off. This method reflects the combination of the first variants of the individual methods.

Consequently, the second variant of the combined approach is based on the second variants of the analytical and DEA approach. By this method, a data point is removed when its deviation from the best fitting analytical cost model is outside of the accepted percentile range for the estimated distribution for this deviation and its cost efficiency is also not in the desired percentile range.

4.2. Machine Learning Model

After the removal of the outliers from the data, the remaining parts should mostly reflect the relationships between the features of the products and their final cost without many influences of other non-product-related parameters. This data can now be used to build a model for the cost prediction. In the proposed approach, this is performed by a ML model. This has the advantage that no — or only limited — further knowledge about the cost relationships is needed. ML models do need substantially less assumptions and prior knowledge than e.g. specialised parametric models, which makes them more flexible. With cleaned data this should also result in an more accurate representation of the actual cost relationships.

To estimate the cost, the ML algorithm learns a function between the inputs (the features of each part) and the output (the cost). Once this function is found, it can be used to predict the output for any other input. Commonly used algorithms for such tasks are SVRs, ensembles of Decision Trees (forests) and MLPs. How each of these methods work in detail is explained in Chapter 2.

The learned ML model contains the functional relationship between the input features and the output of the historical data, which can be used for the cost estimation of future parts. This also means that only the features for the part in question are needed to perform the cost estimation and no additional knowledge. This simplicity in the application of the cost model is a major advantage of the proposed approach.

5. Evaluation of Approaches

For an assessment of the performance of the proposed approach and the different outlier removal methods, a case study on a real-world data set was performed. This chapter gives an overview of the structure of this data, describes the concrete implementations of the proposed concepts and compares their results.

5.1. Data for Case Study

The data used in the case study is a real-world data set of 209 deep-drawn sheet-metal shells collected over a time frame of three years. Due to the confidentiality of the used data, it cannot be shared publicly, but the structure of the data can be seen in Table 5.1. For each part, the four cost drivers net weight m_{net}, complexity of the geometry, material and yearly production volume are given. Additionally, the price payed for each of them is known, which reflects their cost c_{total}. The complexity parameter describes the complexity of the geometry of the part in a numerical way. One stands for simple shapes and two for more complicated ones, for which even multiple stages of deep-drawing are possibly needed. The material grade parameter lists the steel grade with the European steel number specified within EN 10027-2.

Index	Net Weight	Complexity	Material Grade	Yearly Volume	Price
1	2.55	2	1.4509	25000	182
2	0.95	2	1.4513	185000	53
3	1.26	1	1.4301	200000	65
4	0.67	2	1.4828	450000	19
5	4.06	1	1.4510	8400	256

Table 5.1.: Exemplary data entries

All of these four cost drivers are known to influence the cost of a part. The net weight is the most influential parameter because it determines to a high degree the material cost c_m (4.2). This can be seen in Table 5.1 as the ordering of the parts for net weight and for price are the same despite the varying other cost drivers. Furthermore, the complexity plays a role on how much gross material m_gross is used. For simple parts, the waste produced by stamping the part in form is lower than for complex parts. Also, the process scrap r_s is higher for parts with a higher complexity. This explains why Part 3 has only a marginally higher price than Part 2 despite the considerable weight increase. The third parameter influencing the material cost of a part is the material grade which defines different prices per weight p_m for the different steel grades. The last parameter, the yearly volume, impacts the process cost per part c_p (4.3) — higher volumes lead to lower process cost per part, which can be seen at the extraordinary low cost of Part 4.

In addition to the historical data, cost breakdowns for 16 parts from different suppliers were used to extract information and build the analytical cost model. These detailed breakdowns contain a split of the cost according to the parameters in (4.1) to (4.3) and were estimated by product cost experts.

5.2. Implementation of Outlier Removal Methods

This section covers the implementation details of the filtering mechanisms proposed in Section 4.1. The first step for the two analytical variants was the simplification of the existing analytical model, since it needs additional inputs that are unknown for the historical data set. This simplification was performed in collaboration with experts for cost estimation and manufacturing processes which led to the identification of three different manufacturing scenarios: two automated processes (A1 and A2), which only vary in the size and force of the hydraulic press used, and one manual process (M1). For each of these scenarios, the profit r_profit, SGA r_SGA, scrap r_s and material yield rates were identified by analysis of the 16 detailed cost breakdowns and discussions with the experts. In the same way, the packaging cost c_pkg and handling rate r_h were estimated. Finally, some rules were set up to capture the physical and economic boundaries of the three processes. The manual process is only reasonable for lower yearly volumes, due to the high amount of manual labour. This is the reason an upper limit for the volume was introduced up to which M1 is considered a viable scenario. The second rule implemented distinguishes the scenarios A1 and A2. Since they only differ in the pressing force, the relevant parameters are the net weight of the part and its complexity. The heavier and more complex the geometry of a part, the more force is needed for deep-drawing. To include this dependency in the analytical model, a threshold for the product of net weight and complexity of a part is used to decide whether the smaller press in A1 or the more powerful one in A2 is used. After this process, a simplified analytical cost model was obtained, which only requires the four input parameters available for the historical data.

The implementation of the outlier removal methods based on this analytical model is straightforward. For the first analytical variant, as shown in Figure 4.2a, the theoretical costs for all manufacturing processes fulfilling the rules are determined for each of the 209 historical parts. Then, for each product, the process with the least absolute deviation from its real price is identified as its most probable manufacturing scenario. After that, the relative deviation of the two prices is calculated according to (4.5). If the absolute value of this deviation d exceeds a set threshold, this product will be removed from the data set.

The second analytical variant is following the same steps as the first one up to the calculation of d (see Figure 4.2b). Then, the parts are grouped based on their most probable scenario into three subsets and for each of them the distribution of d is estimated. The estimation can be done by using the EDF or by fitting a distribution to the data. For this approach, the latter was chosen as this yields the advantage of generalizing and smoothing of the data, which is especially useful for smaller subsets. The fitting was implemented with the *Python*-package *scipy.stats* [71], which uses a maximum likelihood estimation for the parameters of a distribution. To find a good fit, seven different types of distributions were fitted to each scenario and then tested with a two-sided KS test. To obtain the KS statistics, the implementation of the *scipy.stats* package was used. Since the KS statistic does not follow the Kolmogorov distribution for distributions fitted to the data, it cannot be used to calculate the p-values and therefore the goodness of fit. Instead, the p-values were estimated by simulation according to the procedure explained in Section 2.5. 2500 new data samples were drawn from the fitted distribution of d and for each, the distribution parameters where estimated again. For all of the obtained 2500 distributions, the KS statistic is calculated and compared

Distribution	A1	A2	M1
Burr (Type III)	0.34	0.40	0.30
Alpha	0.16	0.06	0.15
Beta	0.16	0.03	0.13
Rayleigh	0.00	0.00	0.00
Log-Normal	0.06	0.01	0.01
Inverse Gaussian	0.04	0.00	0.00
Gaussian	0.00	0.00	0.00

Table 5.2.: p-values of Kolmogorov-Smirnov tests for distributions fitted to the relative deviation d for each manufacturing scenario

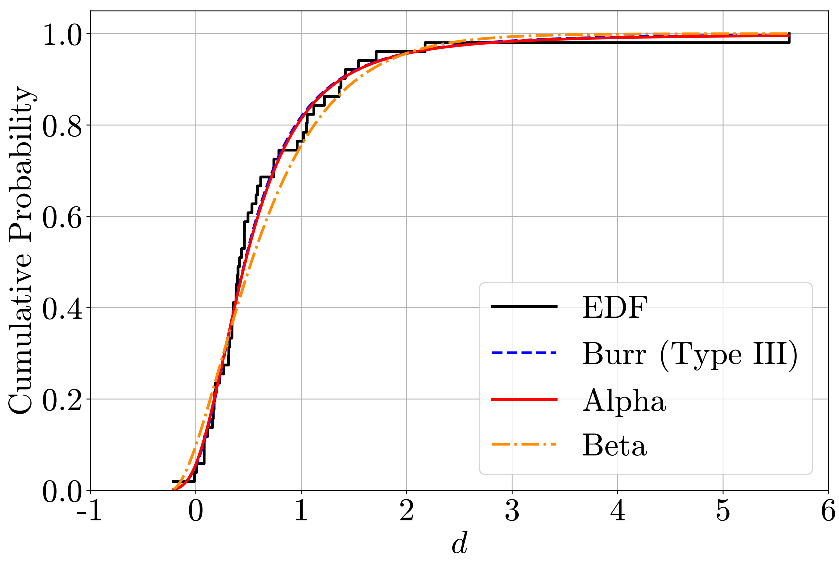


Figure 5.1.: EDF and CDFs of fitted distributions for scenario A1

to the one from the original distribution. The estimated p-value is then the fraction of times the statistic of the new distribution is larger than the one from the original distribution (see Algorithm 2). Based on the rule of thumb used by Clauset, Shalizi and Newman [16], a generation of 2500 data sets will result in a p-value accurate to two decimal digits.

Table 5.2 shows the p-values of the used distributions for the three different scenarios. Since the null hypothesis of the test assumes that the data is drawn

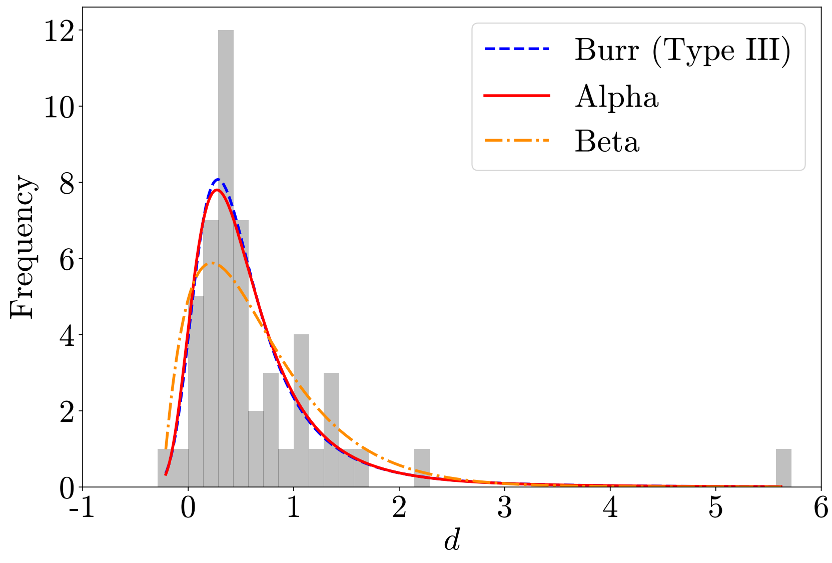


Figure 5.2.: Actual *(gray)* vs. expected frequencies from fitted distributions for scenario A1

from the tested distribution, weaker evidence against the null hypothesis is preferred. Therefore, higher p-values represent a "better" fit. If a conservative choice of $p \leq 0.1$ to rule out the null hypothesis is applied, the Burr distribution cannot be rejected for any reference process. For A1 and M1 d could also follow an alpha or beta distribution, whereas for A2 only the Burr distribution remains. To illustrate the goodness of fit, Figure 5.1 depicts the EDF and the CDFs of the three distributions and Figure 5.2 shows the histogram of the actual data compared to the expected frequencies. The same plots for the other scenarios can be found in Appendix B Figure B.1 to B.4. It can be seen that the difference between the Burr and alpha distribution is marginal compared to the difference in the p-values. Nevertheless, the Burr distribution seems to fit the data slightly better according to the plots. Because of this and the fact that it is not rejected for all three reference processes based on the p-values, it was chosen to model the distribution of d for all of these. Any part on the edges of these distributions were considered outliers. To remove them, a percentile range that contains all the non-outliers was defined. It was set up in a way that the same percentage of data was removed from the upper and the lower ends of the distributions, e.g. from the fifth to the 95th percentile.

Material Grade	Numerical Encoding
1.4512	0
1.4510	1
1.4301	2
1.4509	3
1.4513	4
1.4828	5

Table 5.3.: Conversion of material grade into numerical values

Because there was no knowledge about the amount of true outliers on the upper and lower end, this symmetric percentile range was used. If there was more knowledge about the location of the outliers, this could be implemented in an asymmetric percentile range.

For the DEA variants, the BBC model as described in (2.26) was implemented with the linear programming library *PuLP* [51]. To apply the DEA on the data from this case study, it had to be transformed into numerical data first. Three of the four features and the target variable were already numerical — only the material grade needed to be converted. As it was an ordinal variable, the material grades were converted to ascending numbers, as seen in Table 5.3, starting with the cheapest material. After this transformation, the DEA was performed. In the first variant, the so calculated efficiencies for each of the 209 parts of the historical data set were then used directly to remove any data sample with a efficiency lower than a certain threshold.

The second DEA variant removes outliers at the lower and upper end of the efficiency distribution. As shown in Figure 5.3, this distribution does not follow a common distribution for this case study and hence, the percentile approach described above was used with the EDF. Again, a symmetrical percentile range was defined and any part with a efficiency outside of this range was removed as an outlier.

Both combination approaches were implemented in putting the already described methods together. The first variant uses the implementations of the first variants of the analytical and DEA approaches to identify outliers. Data points are only removed for the training data when both approaches flag them as outliers. This means, any part with an absolute d-value higher then the set

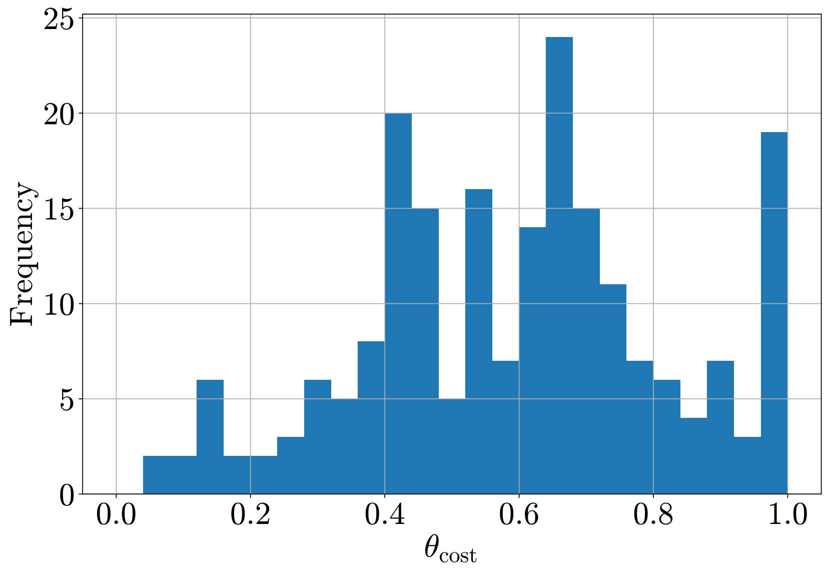


Figure 5.3.: Histogram of cost efficiencies

threshold and a cost efficiency lower than the acceptable one are not part of the training data set.

Similarly, the second variant combines the other two variants of the approaches — again only removing parts that are identified as outliers by both. This results in any part whose deviation from the analytical model and cost efficiency are not in the accepted percentile ranges of the estimated distributions being deleted.

5.3. Building of Machine Learning Models

In the final step the filtered data was used to train ML models. To ensure that the differences between the outlier removal methods are not just a product of the combination with a particular model, four different models were used to evaluate the approaches: two SVRs, one with a linear kernel and the other with a RBF kernel, an AdaBoost Forest and a MLP. The linear SVR is the simplest model, which can only represent linear relationships. The other three models have a higher expressive power and can fit more complex functions to the data. Moreover, this selection covers deterministic (SVR) and stochastic models (AdaBoost Forest and MLP).

Before the historical data could be used for any ML model, it had to be preprocessed. First the material grade had to be converted to a numerical scale. Therefore, the same method mentioned earlier and shown in Table 5.3 was used. After that, each input feature and the output were standardised by subtracting the mean and dividing by the standard deviation. This results in similar scales for all features and is necessary as SVRs are not scale invariant and MLPs tend to converge faster with standardised data.

For the implementation of the SVRs, the *SVR* class of the *scikit-learn* [56] library was used — once with a linear and once with the RBF kernel — which describes the ϵ-SVR covered in Section 2.1. The AdaBoost Forest was built with the *AdaBoostRegressor* class which used a *DecisionTreeRegressor* as base estimator. The MLP, on the other hand, was implemented with the *Sequential* model from the *Keras* [14] library integrated in *TensorFlow* [1] with two fully connected hidden layers. To incorporate the Dropout, additionally after each hidden layer a Dropout layer was added.

To find a good set of hyperparameters, for all four models a grid search with different parameter settings was performed on the unfiltered data set. Since a single split of the data into training and test data introduces randomness in this process, each setting was tested with a ten-fold cross validation. This means, the data was split in ten subsets and nine of them are used for the training of the model. The left-over subset is then used to evaluate the performance. The procedure is then repeated until all subsets were used for testing. After that, the MSE for all test sets was averaged and the standard deviation was calculated to identify the best parameter setting. This was considered a reasonable compromise between computation time and robustness of the results.

Table 5.4 presents the tested hyperparameters for each model. The number of values tested for each parameter was limited to three to five to keep the number of combinations in a reasonable range. The hyperparameters for the linear SVR are the regularization parameters C and ϵ. For C five values between 0.1 and 500 were tested, whereas for ϵ three values from 0.01 to 0.1 were identified. Here, the upper limit marks the maximum tolerated deviation seen as acceptable for this application.

The RBF variant of the SVR has the additional kernel parameter γ. A commonly used rule of thumb for the γ-value is to take the inverse of the product

Model	Hyperparameter	Values
Linear SVR	C	0.1, 1, 10, 100, 500
	ϵ	0.01, 0.05, 0.1
RBF-SVR	C	0.1, 1, 10, 100, 500
	ϵ	0.01, 0.05, 0.1
	γ	0.025, 0.25, 2.5
AdaBoost Forest	Min Samples	1, 2, 4, 5
	Number of Estimators	5, 10, 50, 100
	Learning Rate α	0.1, 0.3, 0.5, 0.8
MLP	Neurons per Layer	5, 8, 10, 12
	Dropout Rate	0.1, 0.2, 0.3
	Batch Size	10, 25, 50
	Number of Epochs	1000, 2000, 3000, 4000

Table 5.4.: Hyperparameter values tested with grid search

of the number of features times the variance of the flattened feature matrix. Since the variances of all features equal one due to the standardisation, this can be simplified to $1/N_{\text{features}}$. As alternatives $10/N_{\text{features}}$ and $1/(10 \cdot N_{\text{features}})$ were chosen to expand the search space to values that are an order of magnitude higher and lower. For both SVRs, minimization was stopped when the result did not improve for at least 0.001 for two consecutive iterations.

The hyperparameters tuned for the AdaBoost Forest were the stopping criterion for the individual Decision Trees, the number of estimators and the learning rate α used for boosting. The training of the trees was stopped as soon as any further split would have led to a leaf node having less samples than a minimum number. This is an efficient and easy-to-tune method to avoid overfitting of Decision Trees. First trials showed a good performance with a minimum number of two to four samples in the leaf nodes and around 50 trees, which was expanded to lower and higher values in the grid search. To measure the quality of a split the MSE was used as described in Section 2.2. Additionally, for the boosting the linear loss was chosen.

One major hyperparameter for the MLP is the amount of neurons in each hidden layer. Here, small values between five and twelve were investigated to avoid potential overfitting. For the same reason, the rate of the Dropout per-

Model	Best Hyperparameter Setting
Linear SVR	$C = 100$; $\epsilon = 0.1$
RBF-SVR	$C = 10$; $\epsilon = 0.05$; $\gamma = 0.25$
AdaBoost Forest	Min Samples = 2; Number of Estimators = 100; $\alpha = 0.5$
MLP	Neurons per Layer = 10; Dropout Rate = 0.1; Batch Size = 10; Number of Epochs = 2000

Table 5.5.: Best hyperparameter settings for each model

formed after each hidden layer was included as an hyperparameter. Compared to commonly used Dropout rates of around 0.5, for this application relatively low values between 0.1 and 0.3 were tested because the small MLPs are already limited in their overfitting capability. Besides, for the batch size and the epochs (iterations) a set of common values were tested. The *ELU* function (see Appendix A Figure A.1) was chosen as the activation function for the neurons as it leads to faster convergences and better generalization performance compared to the commonly used *ReLU* [17]. For the training itself, the widespread *Adam* optimizer [39] was used, which is a variant of the classical stochastic gradient descent using variable learning rates for each weight.

The best hyperparameter settings for each model can be seen in Table 5.5. The detailed results for the five best settings for each model can be found in Appendix C Table C.1 – C.4. For the SVRs, the best settings were easily identifiable as the ones with the lowest average error also had the lowest standard deviation. For the AdaBoost Forest and the MLP, on the other hand, the hyperparameters with the second lowest average MSE were chosen, since these settings are only marginally worse regarding the error but have a lower standard deviation, which indicates better generalization performance.

After obtaining the best hyperparameter setting for each model, these were implemented in the cost estimation approach. First, the historical data set was filtered with one of the outlier removal methods. Then, the reduced data was standardised and all four ML models were tested via a 50-fold cross validation. 50 folds were chosen as a compromise between computation time for the training of the models and statistically sound results. To avoid any leakage of information from the training to the validation set, the standardisation was performed after each split of the data into the two sets, instead of performing it on the whole data set in the beginning. This means, each validation set was standardised with the mean and standard deviation from the corresponding training set. In addition to the normal validation, for each iteration of the cross validation, the fitted model was also tested on the whole data set including the outliers removed before training.

5.4. Performance of Approaches

In order to compare the different outlier removal approaches, each method was tested with different thresholds of the decision parameter. The accuracy of the resulting four ML models for each method was used to investigate the behaviour of the approaches in relation to the amount of data classified as outliers. As a measure of accuracy, the Root Mean Squared Error (RMSE) was used since the models had been trained based on the MSE, which can be difficult to interpret because of its quadratic nature. The RMSE negates this problem while still stronger penalising larger errors in contrast to alternatives like the Mean Absolute Error. The thresholds and percentages of outliers removed, as shown in Table 5.6, are chosen such that each stage leads to a comparable amount of outliers. For both combination variants, all combinations of the listed values were investigated. To achieve a similar amount of identified outliers with them, the threshold values also had to be raised, since the outliers which are found by both methods are only a subset of the outliers flagged by each individual approach.

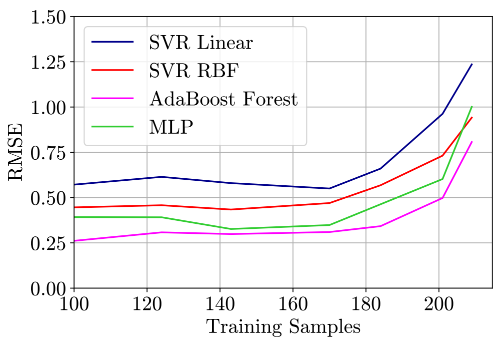
Outlier Removal Approach	Threshold/Outlier Percentage
Analytical Variant 1	0.2, 0.3, 0.4, 0.6, 0.8, 2.0, ∞
Analytical Variant 2	50, 40, 30, 20, 10, 2, 0
DEA Variant 1	0.60, 0.55, 0.50, 0.40, 0.30, 0.20, 0.00
DEA Variant 2	50, 40, 30, 20, 10, 2, 0
Combination Variant 1	$\lvert d \rvert$: 0.15, 0.45, 0.85, ∞
	θ_{cost}: 0.65, 0.55, 0.35, 0.00
Combination Variant 2	d: 70, 50, 30, 0
	θ_{cost}: 70, 50, 30, 0

Table 5.6.: Thresholds and outlier percentages used for the evaluation of the outlier removal approaches

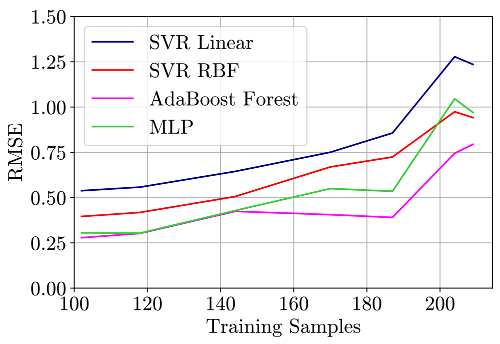
5.4.1. Evaluation via Cross Validation

To evaluate the performance, the results of the cross validation are discussed first. Looking at Figure 5.4, the average RMSE of the 50 cross-validation runs for the different thresholds can be seen. With each approach the models perform according to their complexity: The best accuracy is achieved by the AdaBoost Forest, followed by the MLP. The third place goes to the SVR with RBF kernel and the linear SVR performs worst. Nonetheless, for the analysis of the different outlier removal methods, the shape of the graphs is more important than the concrete performance of a particular model. The first analytical variant starts with a steep decline of the error moving from 209 training samples — representing the whole data set — to 201. The removal of only eight parts leads to a decrease of the error of around 31 % averaged over the four models. The decrease is quite similar between the linear SVR, the MLP and the AdaBoost Forest; only the performance of the RBF-SVR benefits less than the others. After that, the decline slows down and stops at 170 samples with an average reduction of 58 % compared to the unfiltered data set. The behaviours of the four ML models share strong similarities, although the overall RMSE differs. Going below 170 training samples seems to have no effect on any of the models, except for some minor oscillations, which are most likely caused by different cross-validation splits. The initial steep decrease of the error, which slows down afterwards, indicates that the first outliers identified deviate the most from the rest of the data. By declaring additional data points as outliers, the deviation between the new outliers and the other data shrinks, which leads to a slower decrease of the RMSE. Interestingly, the error is not decreasing below 170 samples. Since the removal of more data removes the variance further, it was expected to see at least a small decrease of the prediction error.

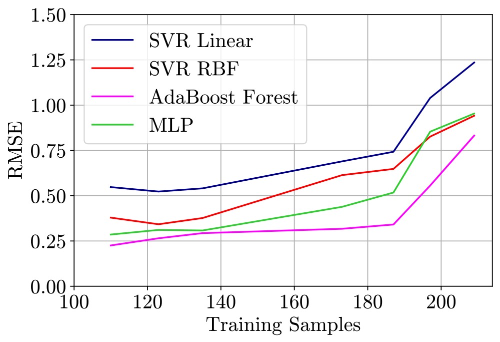
Instead of looking at the average of the 50 cross-validation runs, they can also be illustrated with boxplots, like in Figure 5.5. The whiskers in these boxplots follow the 1.5-interquartile-range-rule and any data exceeding these limits is not shown for the sake of clarity. Here, it can also be seen that the variance between the runs reduces from 209 to 170 training samples. From there on, any further removal of outliers does not change the boxplots considerably. The reduction in variance for the cross-validation runs implies a simultaneous reduction of the variance in the underlying data, explaining the initial decrease of the RMSE.



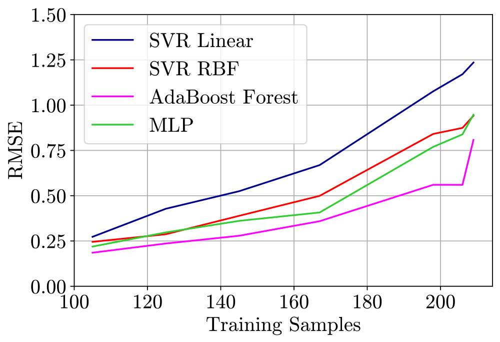
(a) Analytical Variant 1



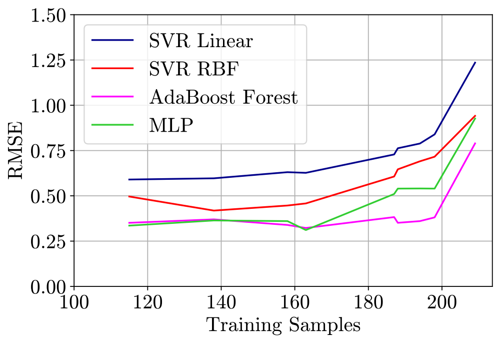
(b) Analytical Variant 2



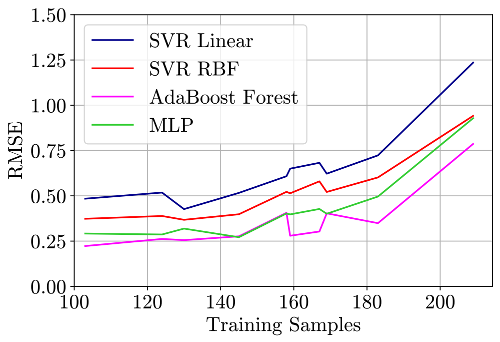
(c) DEA Variant 1



(d) DEA Variant 2

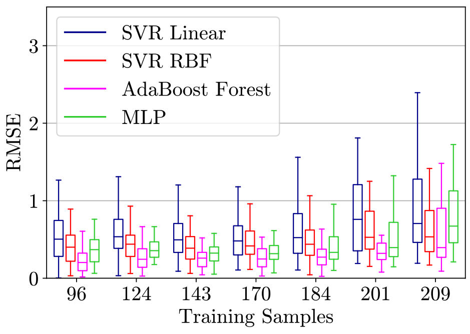


(e) Combination Variant 1

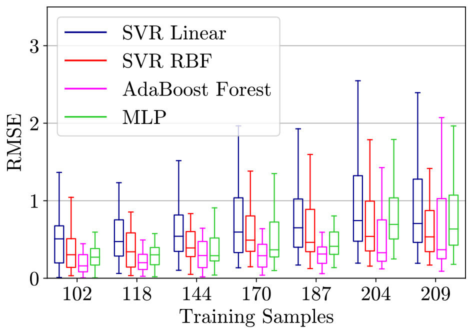


(f) Combination Variant 2

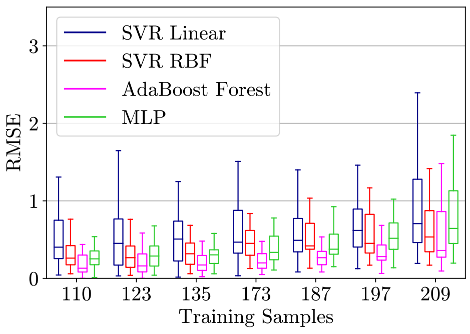
Figure 5.4.: Average RMSE from the cross validation in relation to the training set size for each outlier removal approach



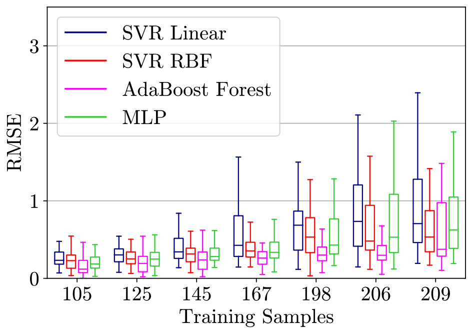
(a) Analytical Variant 1



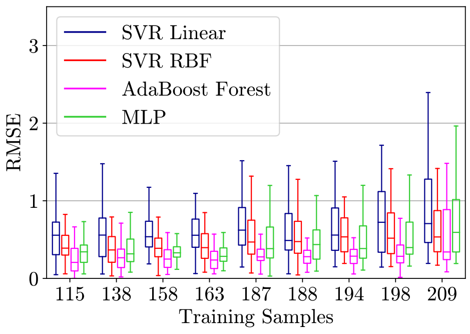
(b) Analytical Variant 2



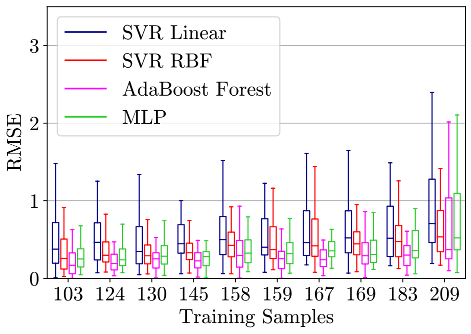
(c) DEA Variant 1



(d) DEA Variant 2



(e) Combination Variant 1



(f) Combination Variant 2

Figure 5.5.: Boxplots of the cross validation RMSE in relation to the training set size for each outlier removal approach

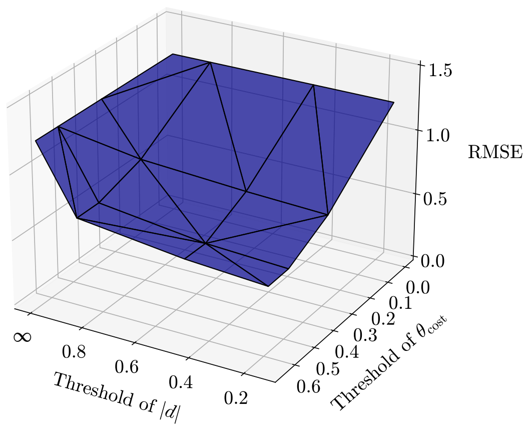
The average error of the second analytical variant has an interesting behaviour when the first outliers are filtered out. Removing five data points from the data set increases the error by 2 % on average. This can indicate that not enough major outliers were removed and the remaining ones are now even harder to predict for the models. This assumption is supported by the following steep decrease of the error, after the removal of additional outliers. At 187 samples the RMSE on average has reduced by 37 % compared to the unfiltered data set. Again, the RBF-SVR shows the slowest decrease. After that, the decrease slows down. The SVRs gradually improve their performance until they end up at similar errors to the ones of the first analytical variant for 100 samples. The AdaBoost Forest and the MLP on the other hand, first plateau for a while until they slowly decrease again. This could indicate that at 187 samples a good model is found, which then only improves, if a certain amount of additional data and with it variance is removed from the training set. Interestingly, the AdaBoost Forest does not suffer from the initial increase of the error and has the longest plateau from 187 to 144 samples. This behaviour could indicate that this model is complex enough to accurately predict even the outliers. Overall, the slower decrease compared to the first method can be explained by the fact, that this approach is not focusing on removing the most obvious outliers, but on maintaining the initial distribution of the deviation from the analytical model. Since data points are removed on both ends of the distribution, the ones on the lower end are less influential then the ones from the upper tail. By this, more variance is left in the data as a trade-off. This can also be seen in the boxplots. The whiskers and the interquartile-range is shrinking slower than in the first variant, but continuously over the whole range, except for the increase in the beginning.

Looking at the DEA approaches, the first variant has a similar behaviour as the second analytical variant. Nevertheless, the decrease up to 187 samples is greater with an average of 44 %. Again, the steepest slope is not at the beginning but between 197 and 187 samples; except for the AdaBoost Forest, whose decrease is constant from 209 to 187. After this, all models slowly increase their performance up to 135 training samples. This improvement is only marginal for the AdaBoost Forest as its error at 187 is already quite low. Looking at the corresponding boxplots the decrease of variance can be seen up to 135 data samples. Below that, almost no improvement is noticeable. Compared to the analytical variants, less variance can be seen in the results. The notable behaviour of the AdaBoost Forest can also be seen in the trend of the variance: A great decrease in the first step, followed by only marginal improvement. When compared to the first analytical variant, this method has similar — even slightly lower — variances around 185 and 200 samples but worse average errors. This could be an indication that this approach indeed reduces the variance but does not follow the underlying relationships in the data set as well.

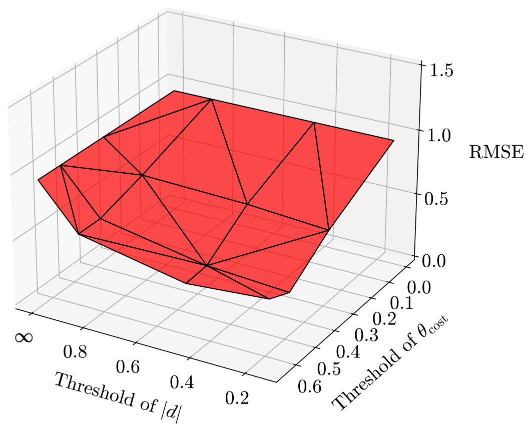
A more distinct behaviour is shown by the second DEA variant. Here, the initial steep decline of the average error is only present for the AdaBoost Forest. This model reduces its RMSE by 31 % by removing only three parts. The other three models decrease their error almost linearly up to 100 samples. There is however a small dent noticeable around 167 training samples. A remarkable difference to the other approaches so far is the decrease of the difference between the worst (linear SVR) and the best model (AdaBoost Forest). The more training samples are removed, the closer the errors of all models get. This could be due to a oversimplification of the problem by this approach. If this method removed any part that is not following a linear relationship, the non-linear models would not have any advantage in performance and a similar convergence of the accuracies of the different models could be seen. A look at the boxplots reinforces this assumption. This approach leads to the lowest variances in the cross-validation runs, even for the linear SVR.

Investigating the results of the combination variants, the first variant looks similar to the underlying approaches. A steep decent of an average of 37 % can be seen for a removal of 11 outliers, which is similar to the first analytical variant. The RBF-SVR shows once more the least initial improvement. The following behaviours look like an average of both approaches. The models slowly improve until they stagnate around 160 samples. The best accuracy they reach is on par with the first analytical variant and worse than the first DEA variant. The corresponding boxplots show that the variance in the data is also in between the two individual approaches.

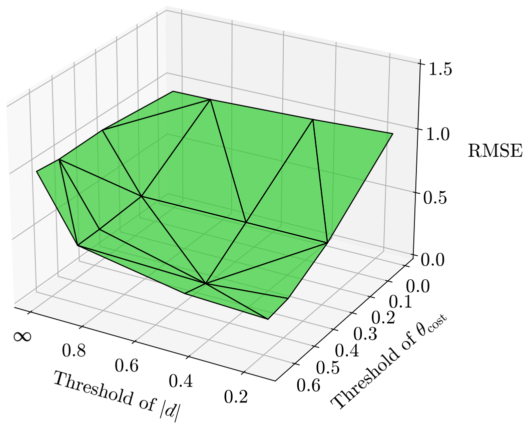
The second combination variant also combines characteristics from both components. A decrease of the RMSE of 45 % can be seen up to 183 samples. After that the decrease slows down and ends up at similar levels as the second analytical variant. The convergence of the models observed in the second DEA variant, is not present, although the decrease of the error is faster as with analytical method. A comparison of the boxplots confirms these results.



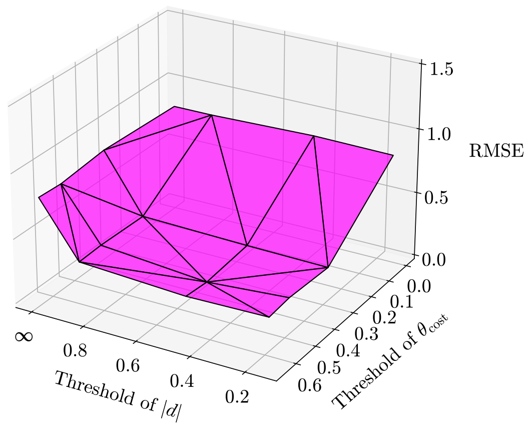
(a) Linear SVR



(b) RBF-SVR



(c) MLP

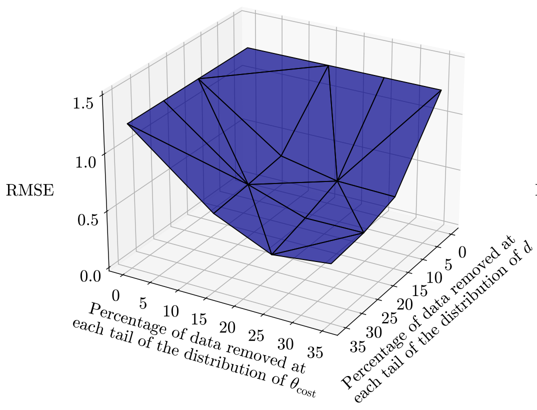


(d) AdaBoost Forest

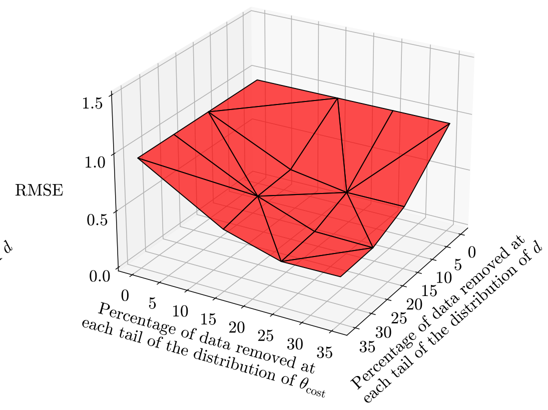
Figure 5.6.: Average RMSE from the cross validation in relation to the thresholds for cost efficiency and deviation from the analytical model for the first combination variant

The initial drop of the variance is similar to the DEA variant, whereas the behaviour at lower samples resembles the one of the analytical method.

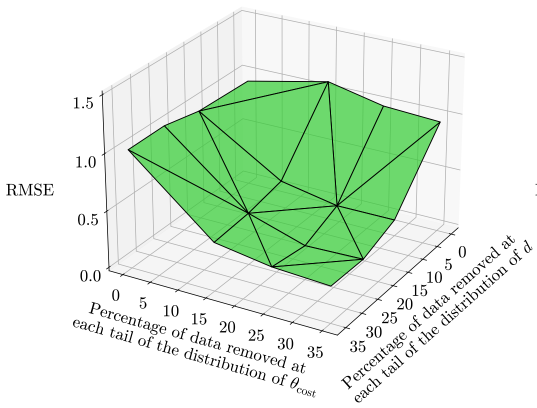
The small peaks in the plots of the combination variants can be explained by how the plots were generated. Since these approaches have two different threshold parameters, they span a three-dimensional space with the RMSE, which is then transformed into a two-dimensional one by plotting the resulting training samples. Therefore, adjacent points on the 2D-plot are not necessarily adjacent in the three-dimensional space and can be from regions where



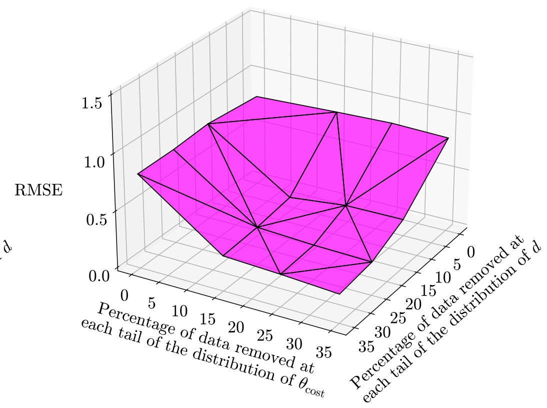
(a) Linear SVR



(b) RBF-SVR



(c) MLP



(d) AdaBoost Forest

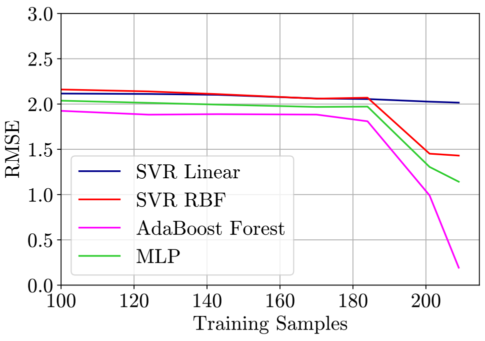
Figure 5.7.: Average RMSE from the cross validation in relation to the thresholds for cost efficiency and deviation from the analytical model for the second combination variant

different approaches dominate the behaviour of the combination approach. Additionally, when multiple threshold combinations led to the same amount of training samples, only the combination with the lower average error was plotted. Looking at the 3-D plots of the first variant (Figure 5.6), it can be seen that the lines on the surface parallel to the axis for the cost efficiency θ_{cost} decrease slowly compared to the lines parallel to the $|d|$-axis. The latter show a steep decrease of the RMSE between the whole data set and a threshold for $|d|$ of 0.8. With lower thresholds for the cost efficiency this decline is weakened. For thresholds lower than 0.8 for $|d|$, the error is only decreasing slowly for all models. Looking at the cost efficiency lines, they are almost not affected by the threshold for $|d|$, except for the range between 0.8 and ∞. All in all no synergies between the two methods can be seen. It rather seems that they hinder each other.

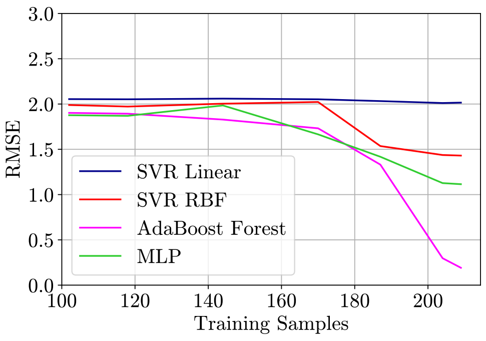
For the second combination variant both thresholds have a similar effect, as seen in Figure 5.7. The lines parallel to the d-axis and the ones parallel to the θ_{cost}-axis both show changes when the other threshold is changed. The general trend of a decrease of the slope is existent for every line, but the intensity of the improvement of the RMSE changes. But similar to the first combination variant, no synergy effects can be seen.

5.4.2. Evaluation on Whole Data Set

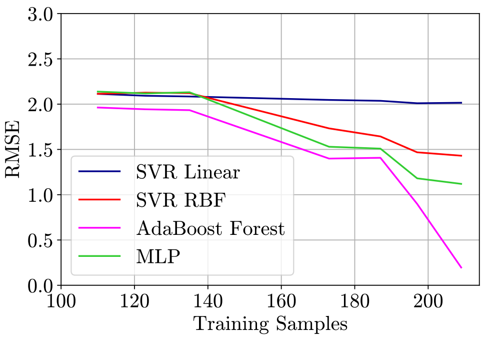
To verify the drawn conclusions from the cross validation, a different look at the approaches might be helpful. The second performance metric used is the RMSE of the trained models applied to the whole data set. This means, the test set contains the outliers that were removed for training in addition to the regular training data. Figure 5.8 shows the average error of the 50 models trained via cross validation for the whole data set. Investigating the first analytical variant, it shows that all models except the linear SVR try to model the outliers when trained on the whole data set. The AdaBoost Forest has a surprisingly low error which indicates that this model is most prone to overfitting. The MLP and RBF-SVR also tend to model the outliers but to a smaller extent. The linear SVR, on the other hand, has a hard time with these data points. Since it can only model linear relationships, it cannot deal with outliers that are probably highly non-linear. As this behaviour is seen in all of the tested approaches, the linear SVR will not be further mentioned in the upcoming analysis. With the removal of more data points, the other models also focus predominantly on the actual underlying relationship, which results in a worse error, since the outliers are predicted worse. For this approach, a fast increase which stops at 184 samples can be seen for all models. The AdaBoost Forest has the highest increase followed by the MLP and RBF-SVR. The last one shows an interesting behaviour as the removal of the first eight outliers increases the error only marginally. Only the removal of 17 more data points brings the error up to its maximum score. A similar behaviour can be seen for the MLP, although not as pronounced. The first few outliers still increase the RMSE, just with a lesser slope. This behaviour indicates that both models are not complex enough to represent all outliers in the data set.



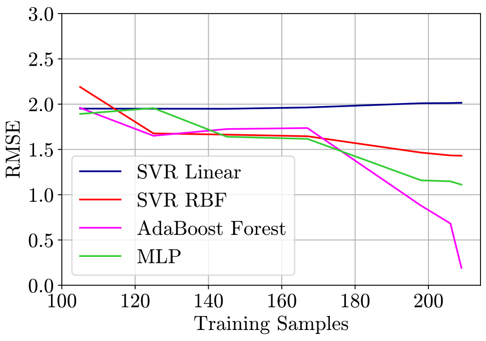
(a) Analytical Variant 1



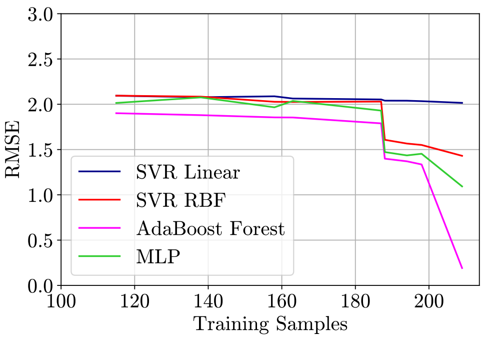
(b) Analytical Variant 2



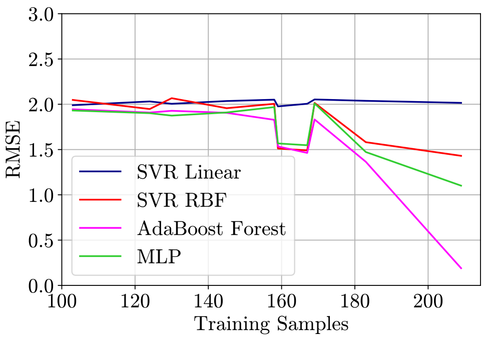
(c) DEA Variant 1



(d) DEA Variant 2



(e) Combination Variant 1



(f) Combination Variant 2

Figure 5.8.: Average RMSE on the whole data set in relation to the training set size for each outlier removal approach

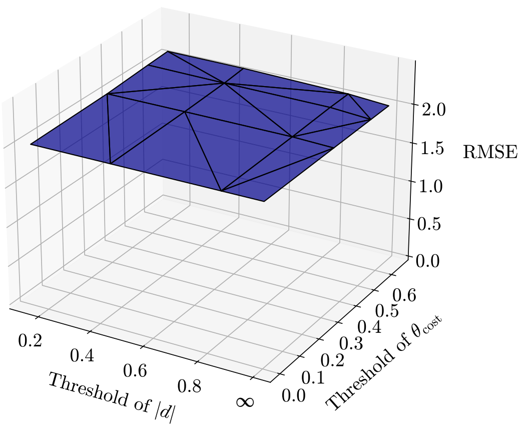
Therefore, the removal of some outliers only results in a slight change of the model. Overall this confirms the observations of the aforementioned evaluation of the cross-validation results.

Looking at the second analytical variant, the most obvious difference is a slower and more gradual increase of the error. Depending on the model this increase stagnates between 170 and 118 training samples. The slow increase for the removal of the first outliers can be seen for all three relevant models. After this, the AdaBoost Forest behaves similar to the first variant, despite a slower increase of the error in general. This results in a late stagnation of the error at 118 samples. The error of the MLP, on the other hand, increases linearly until the maximum is reached at 144 samples. For the RBF-SVR, the initial slow increase is extended up to 187 samples. With the removal of 17 more outliers it already reaches its maximum error. These behaviours show again, that this method is less efficient than the first analytical variant in removing the major outliers.

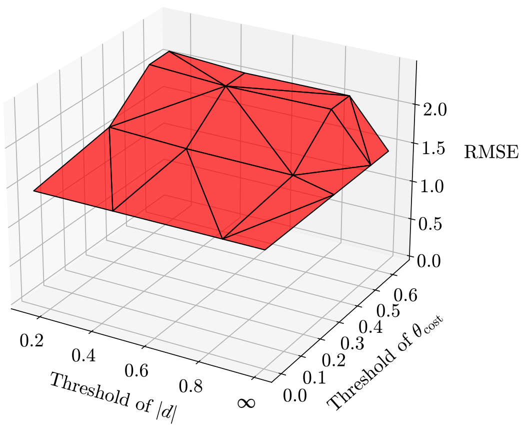
For the first DEA variant, the slow initial increase can be seen again for the MLP and the RBF-SVR. Only the AdaBoost Forest shows a steep rise of the RMSE up to 187 samples, where it then plateaus until 173 samples are reached. With the removal of more data points, the error is then increased again until the maximum is reached at 135 training samples. The MLP shows a similar behaviour. Even the error of the RBF-SVR slows down it rise at 187 samples. The plateaus and the slower increase of the error suggests that this approach removes many data points that have no influence on the models. Consequently, it does not fully capture the underlying relationships in the data. The same assumption was already made in the first analysis and seems now even more plausible.

The plot of the second DEA variant shows a similar picture. The errors of the MLP and RBF-SVR rise slowly up to 167 training samples. The increase for the AdaBoost Forest is again much steeper. The RBF-SVR and the AdaBoost Forest plateau from 167 to 125 samples, after which they jump to their final RMSEs. The plateau of the MLP is a bit shorter, such that it is reaching its maximum error at 125 samples. This approach seems to fail at reliably removing the outliers even more than the first DEA variant. Together with the results of the first evaluation it seems to be the worst approach so far.

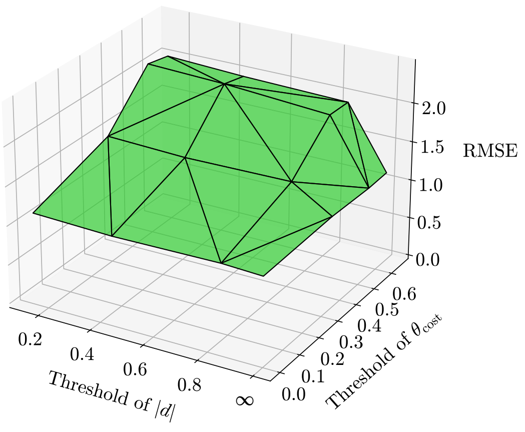
The plots of the combination variants do not reveal much new information. Both variants show again characteristics of both individual methods. For the



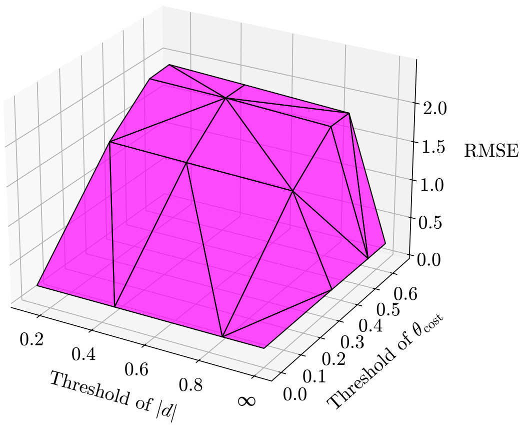
(a) Linear SVR



(b) RBF-SVR



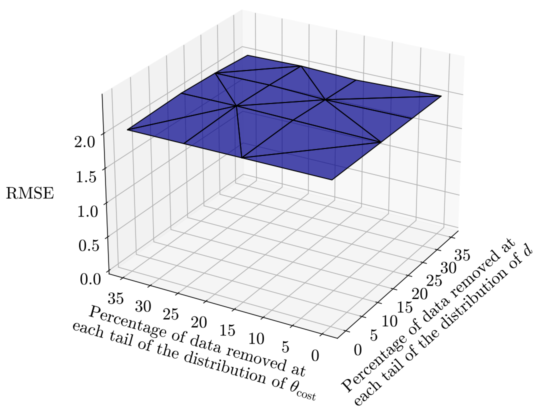
(c) MLP



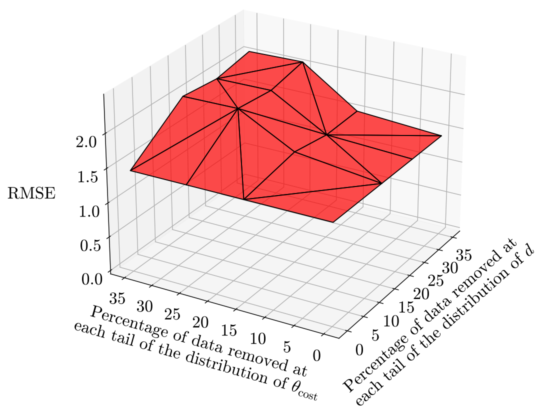
(d) AdaBoost Forest

Figure 5.9.: Average RMSE on the whole data set in relation to the thresholds for cost efficiency and deviation from the analytical model for the first combination variant

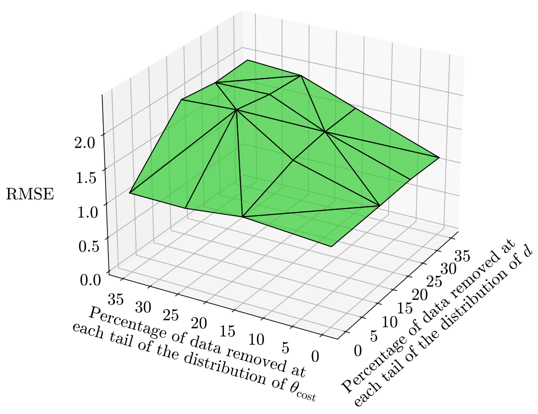
first variant all relevant models have a similar initial rise as in the analytical approach, which is followed by a plateau reminiscent of the first DEA variant. After this plateau all errors rise strongly again and already start to stagnate at 187 samples. Further analysis of this plot is difficult because of the transformation of the three-dimensional results into a 2-D plot. The 3-D plots in Figure 5.9 show the RMSE of the first combination variant. The surface plots show that for all models except for the linear SVR, the performance is limited by the threshold for the cost efficiency θ_{cost}. Since the analytical variant already identified all relevant outliers with a threshold off 0.8, any lower



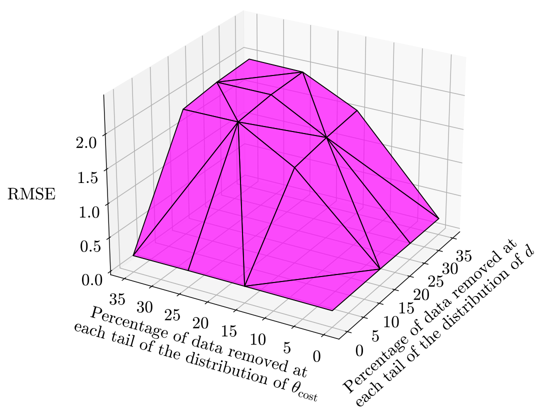
(a) Linear SVR



(b) RBF-SVR



(c) MLP



(d) AdaBoost Forest

Figure 5.10.: Average RMSE on the whole data set in relation to the thresholds for cost efficiency and deviation from the analytical model for the second combination variant

value does not change the performance of the models. The RMSE is then only dependent on how many of these outliers are also found by the DEA.

The plot for the second combination variant in Figure 5.8 shows a rise of the error after the removal of the first outliers similar to the second analytical variant up to 169 samples. After that the RMSE drops and plateaus until 159 training samples, where it is jumping up again, followed by stagnation. This sudden drop is definitely caused by the transformation of the data. Looking at the 3-D plots in Figure 5.10, a similar behaviour as in the first combination variant can be seen. The analytical variant reaches the maximum error way faster than the DEA approach and is therefore limited when smaller percentages of the data are removed based on their cost efficiency.

The boxplots for the evaluation of the performance on the whole data set do not deliver any new insights, since the differences between the 50 cross-validation runs is marginal (ref. Appendix D Figure D.1).

5.4.3. Summary

Overall, it can be concluded that the analytical variants perform best in these evaluations. Both seem to capture the relationships in the data satisfactorily. Which variant is better for the hybrid cost estimation approach, depends on the goal and incentives for the concrete application. The first variant removes outliers more quickly and is therefore suited for scenarios where the fast removal of the major outliers has priority. The second variant is focusing on maintaining the distribution of the historical data, resulting in a slower improvement. But this variant could be beneficial in data sets in which there is no clear line between outliers and inliers, as it gives precise control about the data points it removes. For the outlier removal methods based on the DEA, it has been shown that only the first variant delivers a reasonably good behaviour. It does not reflect the underlying relationship as well as the analytical variants, but for cases where cost efficiency is the main focus, it can be still a valid option. The second DEA variant is neither following the cost relationships nor focuses on efficiency, because the most efficient parts are also removed. These reasons make this approach inferior to the already mentioned ones. Similarly, the combination approaches do not add any benefit to the individual methods. Instead, the combination tends to hinder both methods from removing outliers efficiently and the fact that two thresholds have to be adjusted increases the complexity unnecessarily. Hence, the combination approaches are not recommended to be used in the hybrid cost estimation approach. A short overview of these results is presented in Table 5.7.

The hybrid cost estimation approach as a whole shows promising results. It is a huge improvement over a ML model without any outlier removal with a decrease of the RMSE up to 58 %. The removal methods based on the analytical cost model seem to work best for a realistic cost estimate, but this approach has the flexibility to even shift the priority to the cost efficiency with the DEA-based method. Its major strengths are the simplicity in application,

Outlier Removal Method	Recommendation	Reason
Analytical Variant 1	Yes	Fast removal of major outliers
Analytical Variant 2	Yes	Focus on distribution of historical data
DEA Variant 1	Conditional	Focus on cost efficiency
DEA Variant 2	No	Oversimplification of the cost relationship
Combination Variant 1	No	No benefits over individual approaches
Combination Variant 2	No	No benefits over individual approaches

Table 5.7.: Recommendations for the outlier removal methods to be used in the hybrid cost estimation approach

which is the same of a stand-alone ML model, and the little information needed for an accurate cost estimation of a product.

6. Conclusion & Future Work

Realistic product cost estimation is a crucial element to maintain and extend competitiveness in globalised markets. Due to this importance, a variety of different techniques and approaches had been proposed in literature, which all have their own strengths and weaknesses and are applicable at different times during the development phase. Methods for early phases in the product design process, like intuitive or analogical techniques, require the least information about the final product, but deliver on the other hand only a rough estimate of the cost. Parametric and non-parametric approaches can predict the cost more precisely, if more knowledge about the product is available. The most accurate results can be achieved by a analytical cost analysis. Nevertheless, this technique requires extensive knowledge about the final product design and the manufacturing process, which makes this method only applicable in late development stages.

In case that the product is manufactured by a supplier, additional hurdles for an accurate cost estimation arise. This scenario introduces further uncertainty, which complicates the estimation. One reason for this uncertainty can be missing information about the manufacturing process, which affects analytical approaches negatively. Another reason are numerous non-product related influences on the negotiated price for the manufacturing of the product. Supplier-specific circumstances, like the available machinery or the profit margin, or even the outcome of the negotiations itself add additional variability to the historical data. This impacts parametric and non-parametric models, which use this data.

To deal with these added difficulties, a new hybrid cost estimation approach has been proposed, which combines different cost estimation techniques. The presented method uses a simplified analytical cost model and/or the cost efficiency to filter out unrealistic data points from the historical data set. Afterwards, the cleaned data is used to train a ML model for the cost prediction. The analytical outlier detection methods substitute the missing information with knowledge obtained from examples with full costing details, whereas the DEA approaches focus on the cost efficiency of the products.

The presented hybrid approach showed a superior performance compared to a traditional ML approach as all proposed outlier removal techniques notably improved the accuracy of the cost estimation obtained from the ML model for the performed case study. However, the techniques showed different behaviours. The analytical outlier removal methods captured the underlying relationship in the data best. The first analytical approach proposed is most suited for situations where the outliers are clearly distinguishable from the rest of the data set. The second analytical variant is oriented at the distribution of the original data and can therefore be used for applications where a clear distinction between out- and inliers is difficult. The investigated outlier removal techniques based on a DEA struggled to identify meaningful outliers. Nonetheless, the first of the two presented DEA approaches can still be beneficial when cost efficiency is the main focus of the cost prediction. The second variant, on the other hand, did not deliver satisfactory results as the efficiency focus is lost in addition to the poor modeling of the cost drivers. The combination of analytical and DEA methods showed no additional benefit over the individual approaches. In contrary, the distinct focuses of the analytical model and the DEA seem to hinder each other when combined. Overall, the proposed cost estimation method showed promising results with the right outlier removal methods and is therefore an improvement over classical techniques, especially in cases with limited information on the product. It is a fast and easy-to-use method, which can be applied without extensive knowledge of the product or its cost structure, which makes it a perfect approach for the comparison of different product variants in early design stages.

Nevertheless, more research is needed to discover the full potential of this approach. Next to additional case studies for validation of the results, a deeper analysis of the behaviour of the different outlier removal methods is suggested. This could be done by applying the hybrid cost estimation method on a data set with labeled outliers to see which outlier detection is more reliable. Similarly, it would be interesting to further analyse the differences and common grounds of outliers identified by the combination of analytical and DEA methods and data points only removed by one of the methods.

Appendices

A. Multilayer Perceptron

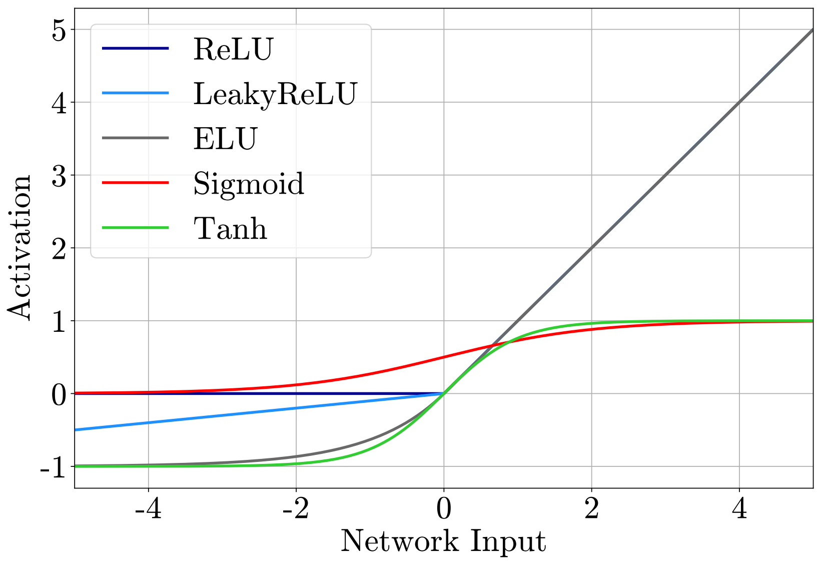


Figure A.1.: Common activation functions in MLPs

B. Fitted Distributions

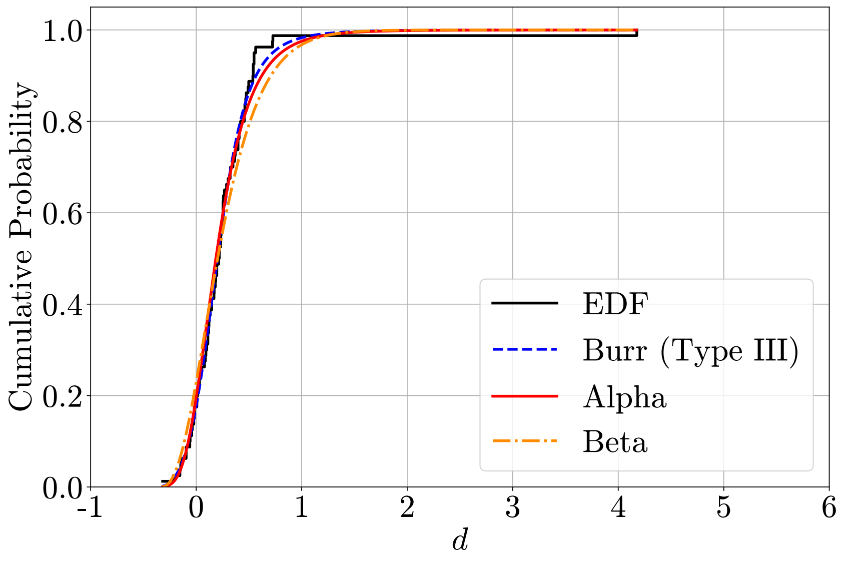


Figure B.1.: EDF and CDFs of fitted distributions for scenario A2

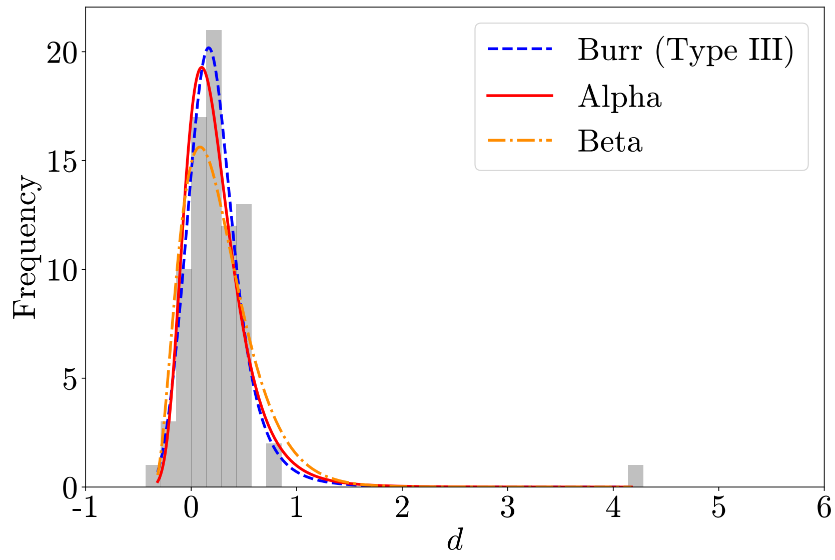


Figure B.2.: Actual *(gray)* vs. expected frequencies from fitted distributions for scenario A2

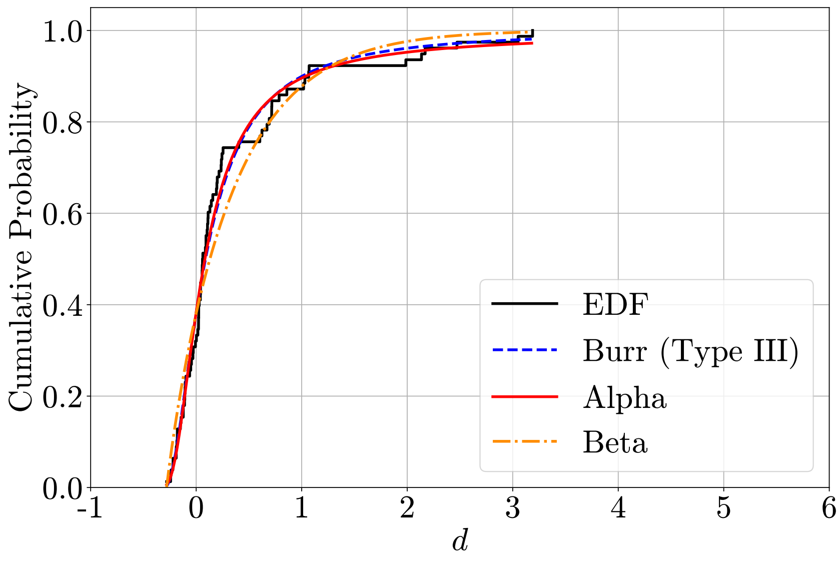


Figure B.3.: EDF and CDFs of fitted distributions for scenario M1

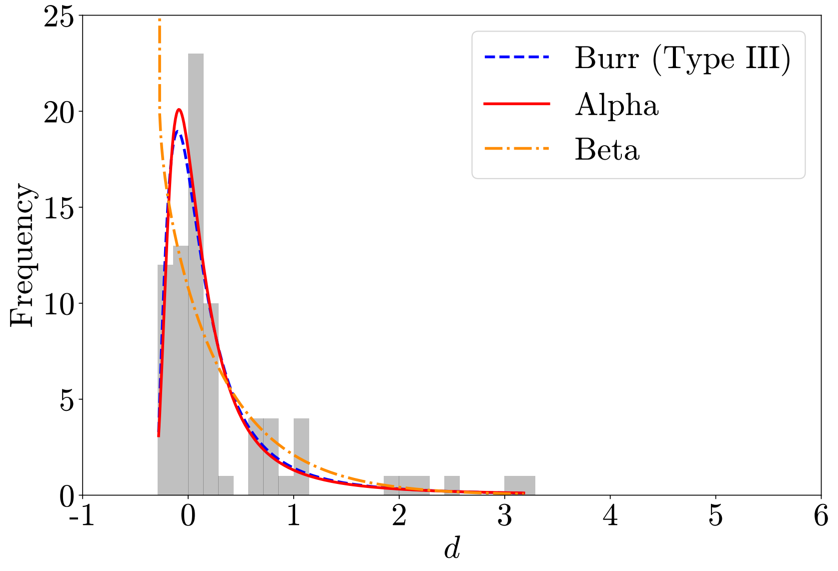


Figure B.4.: Actual *(gray)* vs. expected frequencies from fitted distributions for scenario M1

Distribution	Standardised Probability Density Function	Domain
Burr (Type III)	$f(x,a,b) = \frac{abx^{-a-1}}{(1+x^{-a})^{b+1}}$	$x \geq 0;\ a,b > 0$
Alpha	$f(x,a) = \frac{1}{x^2\Phi(a)\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{a-1}{x}\right)^2\right)$	$x,a > 0$
Beta	$f(x,a,b) = \frac{\Gamma(a+b)}{\Gamma(a)+\Gamma(b)} x^{a-1}\left(1-x\right)^{b-1}$	$0 \leq x \leq 1;\ a,b > 0$
Rayleigh	$f(x) = x\exp\left(-x^2/2\right)$	$x \geq 0$
Log-Normal	$f(x,a) = \frac{1}{ax\sqrt{2\pi}} \exp\left(-\frac{\log^2(x)}{2s^2}\right)$	$x,s > 0$
Inverse Gaussian	$f(x,\mu) = \frac{1}{1\sqrt{2\pi x^3}} \exp\left(-\frac{(x-\mu)^2}{2x\mu^2}\right)$	$x \geq 0;\ \mu > 0$
Gaussian	$f(x) = \frac{\exp\left(-x^2/2\right)}{\sqrt{2\pi}}$	$x \in \mathbb{R}$

where $\Phi(z)$ is the Gaussian CDF and $\Gamma(z)$ is the gamma function

Table B.1.: Probability density functions used to describe the relative deviation d

C. Best Hyperparameter Settings

C	ϵ	Avg. MSE	Std. Deviation MSE
100	0.10	0.53163	0.69001
10	0.10	0.53164	0.69003
500	0.10	0.53177	0.69023
1	0.10	0.53186	0.68990
500	0.05	0.53329	0.69945

Table C.1.: Best hyperparameter settings for the linear SVR

C	ϵ	γ	Avg. MSE	Std. Deviation MSE
10	0.05	0.25	0.33535	0.38772
10	0.10	0.25	0.33987	0.39201
10	0.01	0.25	0.34387	0.40425
100	0.10	0.25	0.38925	0.43652
1	0.10	2.50	0.42139	0.49604

Table C.2.: Best hyperparameter settings for the RBF-SVR

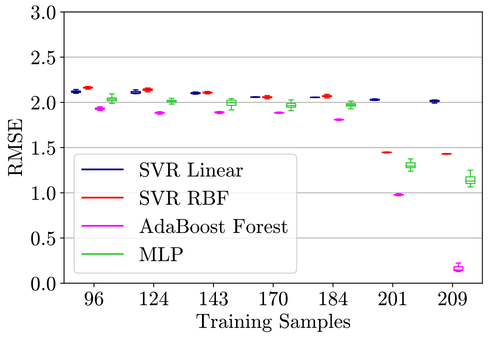
Neurons per Layer	Dropout Rate	Batch Size	Epochs	Avg. MSE	Std. Deviation MSE
5	0.2	25	3000	0.39595	0.48660
10	0.1	10	2000	0.39787	0.44720
8	0.2	50	3000	0.39887	0.46048
8	0.1	10	1000	0.39964	0.47397
5	0.2	50	4000	0.40098	0.45624

Table C.3.: Best hyperparameter settings for the MLP

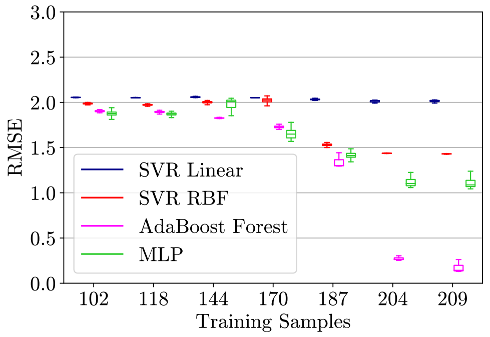
Min Samples	Estimators	α	Avg. MSE	Std. Deviation MSE
1	50	0.5	0.31029	0.19684
2	100	0.5	0.31340	0.15183
2	100	0.3	0.32110	0.16141
2	10	0.5	0.32271	0.21309
4	10	0.8	0.32388	0.24082

Table C.4.: Best hyperparameter settings for the AdaBoost Forest

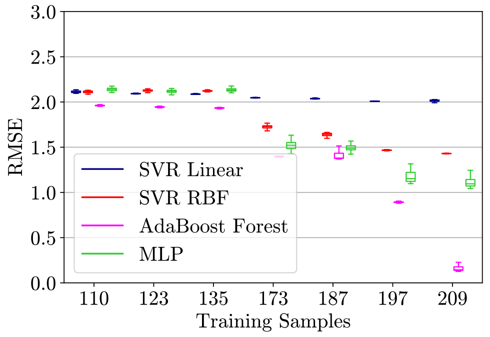
D. Evaluation Plots



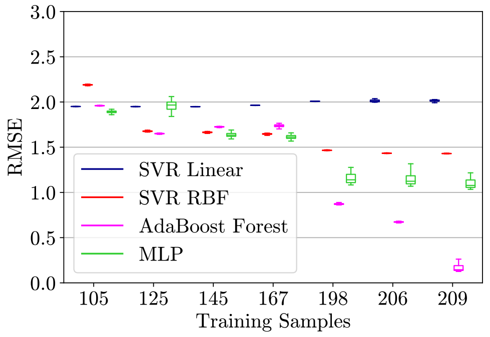
(a) Analytical Variant 1



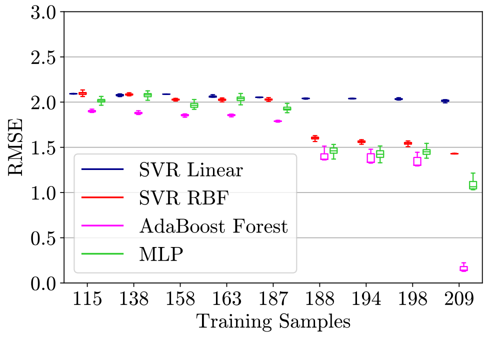
(b) Analytical Variant 2



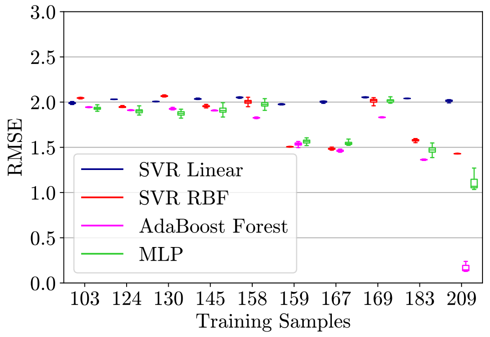
(c) DEA Variant 1



(d) DEA Variant 2



(e) Combination Variant 1



(f) Combination Variant 2

Figure D.1.: Boxplots of the RMSE on the whole data set in relation to the training set size for each outlier removal approach

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I.Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[2] C. C. Aggarwal. *Outlier Analysis*. Springer International Publishing, 2nd edition, 2017.

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. *ACM Sigmod Record*, 28(2):49–60, 1999.

[4] S. Aram, C. Eastman, and J. Beetz. Qualitative and Quantitative Cost Estimation: A Methodology Analysis. In *Computing in Civil and Building Engineering, Joint CIB W78 and ICCCBE Conference, 23-25 June 2014, Orlando, Florida*, pages 381–389. American Society of Civil Engineers, 2014.

[5] R. D. Banker, A. Charnes, and W. W. Cooper. Some Models for Estimating Technical and Scale Inefficiencies in Data Envelopment Analysis. *Management Science*, 30(9):1078–1092, 1984.

[6] R. Bansal, N. Gaur, and S. N. Singh. Outlier Detection: Applications and Techniques in Data Mining. In *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pages 373–377, 2016.

[7] L. Breiman. *Classification and Regression Trees*. CRC Press, 2017.

[8] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. *ACM Sigmod Record*, 29(2):93–104, 2000.

[9] H. Byungho and C. Sungzoon. Characteristics of Autoassociative MLP as a Novelty Detector. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings*, volume 5, pages 3086–3091, 1999.

[10] C. Caroni. Outlier Detection by Robust Principal Components Analysis. *Communications in Statistics - Simulation and Computation*, 29(1):139–151, 2000.

[11] S. Cavalieri, P. Maccarrone, and R. Pinto. Parametric vs. neural network models for the estimation of production costs: A case study in the automotive industry. *International Journal of Production Economics*, 91(2):165–177, 2004.

[12] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

[13] A. Charnes, W. W. Cooper, and E. Rhodes. Measuring the efficiency of decision making units. *European Journal of Operational Research*, 2(6):429–444, 1978.

[14] François Chollet et al. Keras, 2015. Software available from keras.io.

[15] R. G. Chougule and B. Ravi. Casting cost estimation in an integrated product and process design environment. *International Journal of Computer Integrated Manufacturing*, 19(7):676–688, 2006.

[16] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-Law Distributions in Empirical Data. *SIAM Review*, 51(4):661–703, Nov 2009.

[17] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[18] W. W. Cooper, L. M. Seiford, and K. Tone. *Data Envelopment Analysis: A Comprehensive Text with Models, Applications, References and DEA-Solver Software.* Springer-Verlag, Berlin, Heidelberg, 2006.

[19] D. Cousineau and S. Chartier. Outliers detection and treatment: a review. *International Journal of Psychological Research*, 3(1):58–67, 2010.

[20] G. Ćwikła and K. Bańczyk. Similarity of Parts Determined by Semantic Networks as the Basis for Manufacturing Cost Estimation. In Á. Herrero, C. Cambra, D. Urda, J. Sedano, H. Quintián, and E. Corchado, editors, *15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020)*, pages 320–330. Springer International Publishing, 2021.

[21] R. B. D'Agostino and M. A. Stephens. *Goodness-of-Fit Techniques*. Marcel Dekker, 1986.

[22] F. Defersha, A. Salam, and N. Bhuiyan. A new approach for product cost estimation using data envelopment analysis. *International Journal of Industrial Engineering Computations*, 3(5):817–828, 2012.

[23] A. Díaz, S. Fernández, L. Guerra, and E. Díaz. Manufacturing Cost Prediction Through Data Mining. In Á. Rocha, M. Paredes-Calderón, and T. Guarda, editors, *Developments and Advances in Defense and Security*, pages 251–258, Singapore, 2020. Springer Singapore.

[24] H. Drucker. Improving Regressors using Boosting Techniques. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 107–115. Morgan Kaufmann Publishers Inc., 1997.

[25] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. N. Vapnik. Support Vector Regression Machines. In M. C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 155–161. MIT Press, 1997.

[26] O. Duran, J. Maciel, and N. Rodriguez. Comparisons between two types of neural networks for manufacturing cost estimation of piping elements. *Expert Systems with Applications*, 39(9):7788–7795, 2012.

[27] H. Dutta, C. Giannella, K. Borne, and H. Kargupta. Distributed Top-K Outlier Detection from Astronomy Catalogs using the DEMAC System. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 473–478. SIAM, 2007.

[28] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Pro-*

ceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pages 226–231. AAAI Press, 1996.

[29] C. Favi, M. Germani, and M. Mandolini. Analytical Cost Estimation Model in High Pressure Die Casting. *Procedia Manufacturing*, 11:526–535, 2017.

[30] Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, ICML'96, pages 148–156. Morgan Kaufmann Publishers Inc., 1996.

[31] J. D. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference*. CRC Press, 5th edition, 2010.

[32] V. Hautamaki, I. Karkkainen, and P. Franti. Outlier Detection Using k-Nearest Neighbour Graph. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 3, pages 430–433, 2004.

[33] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[34] M. Hubert, M. Debruyne, and P. J. Rousseeuw. Minimum Covariance Determinant and Extensions. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10:e:1421, 2018.

[35] M.F Jiang, S.S Tseng, and C.M Su. Two-phase clustering process for outliers detection. *Pattern Recognition Letters*, 22(6):691–700, 2001.

[36] T. Johnson, I. Kwok, and R. Ng. Fast Computation of 2-Dimensional Depth Contours. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 224–228. AAAI Press, 1998.

[37] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag New York, 2nd edition, 2002.

[38] M. Juszczyk. The Challenges of Nonparametric Cost Estimation of Construction Works with the use of Artificial Intelligence Tools. *Procedia Engineering*, 196:415–422, 2017.

[39] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2014.

[40] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 392–403. Morgan Kaufmann Publishers Inc., 1998.

[41] A. Kolmogorov. Sulla Determinazione Empirica di Una Legge di Distribuzione. *Giornale dell'Istituto Italiano degli Attuari*, 4:83–91, 1933.

[42] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher. *Computational Intelligence: A Methodological Introduction.* Texts in Computer Science. Springer-Verlag London, 2nd edition, 2016.

[43] J. Laurikkala, M. Juhola, and E. Kentala. Informal Identification of Outliers in Medical Data. *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pages 20–24, 2000.

[44] Z. Leszczyński and T. Jasiński. An artificial neural networks approach to product cost estimation. the case study for electric motor. *Informatyka Ekonomiczna*, 1:72–84, 2018.

[45] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

[46] J.-L. Loyer, E. Henriques, M. Fontul, and S. Wiseall. Comparison of Machine Learning methods applied to the estimation of manufacturing cost of jet engine components. *International Journal of Production Economics*, 178:109–119, 2016.

[47] M. Mandolini, C. Favi, M. Germani, M. Marconi, and R. Raffaeli. An Analytical Cost Estimation Approach for Generic Sheet Metal 3D Models. *Computer-Aided Design and Applications*, 16:936–950, 2019.

[48] L. M. Manevitz and M. Yousef. One-Class SVMs for Document Classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[49] I. Martinelli, F. Campi, E. Checcacci, G. M. Lo Presti, F. Pescatori, A. Pumo, and M. Germani. Cost Estimation Method for Gas Turbine in Conceptual Design Phase. *Procedia CIRP*, 84:650–655, 2019.

[50] W. S. McCulloch and W. Pitts. A Logical Calculus of the Idea Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[51] S. Mitchell, M. O'Sullivan, and I. Dunning. PuLP: A Linear Programming Toolkit for Python, 2011. Software available from pypi.org/project/PuLP.

[52] J. Mrozinski, M. Saing, M. Hooke, A. Lumnah, and J. Johnson. COMPACT KNN: Developing an Analogy-Based Cost Estimation Model for CubeSats Jet Propulsion Laboratory, California Institute of Technology Pasadena, CA 91109. In *2020 IEEE Aerospace Conference*, pages 1–9, 2020.

[53] A. Niazi, J. S. Dai, S. Balabani, and L. Seneviratne. Product Cost Estimation: Technique Classification and Methodology Review. *Journal of Manufacturing Science and Engineering*, 128(2):563–575, 2005.

[54] F. Ning, Y. Shi, M. Cai, W. Xu, and X. Zhang. Manufacturing cost estimation based on the machining process and deep-learning method. *Journal of Manufacturing Systems*, 56:11–22, 2020.

[55] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. LOCI: Fast Outlier Detection Using the Local Correlation Integral. In *Proceedings 19th International Conference on Data Engineering*, pages 315–326, 2003.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Software available from scikit-learn.org.

[57] T. Petsche, A. Marcantonio, C. Darken, S. J. Hanson, G. M. Kuhn, and I. Santoso. A Neural Network Autoassociator for Induction Motor Failure Prediction. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, pages 924–930. MIT Press, 1995.

[58] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, 65(6):65–386, 1958.

[59] P. J. Rousseeuw and M. Hubert. Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):73–79, 2011.

[60] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, 2005.

[61] I. Ruts and P. J. Rousseeuw. Computing Depth Contours of Bivariate Point Clouds. *Computational Statistics & Data Analysis*, 23(1):153–168, 1996.

[62] N. Sajadfar and Y. Ma. A hybrid cost estimation framework based on feature-oriented data mining approach. *Advanced Engineering Informatics*, 29(3):633–647, 2015.

[63] R. E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5(2):197–227, 1990.

[64] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support Vector Method for Novelty Detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 582–588. MIT Press, 1999.

[65] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. Principal Component-based Anomaly Detection Scheme. In T. Young Lin, S. Ohsuga, C.-J. Liau, and X. Hu, editors, *Foundations and Novel Approaches in Data Mining*, pages 311–329. Springer Berlin Heidelberg, 2006.

[66] N. V. Smirnov. Estimate of deviation between empirical distribution functions in two independent samples [in Russian]. *Bulletin of Moscow University*, 2:3–16, 1939.

[67] N. N. R. Ranga Suri, N. Murty, and G. Athithan. *Outlier Detection: Techniques and Applications.* Springer International Publishing, 1st edition, 2019.

[68] B. B. Thompson, R. J. Marks, J. J. Choi, M. A. El-Sharkawi, M.-Y. Huang, and C. Bunje. Implicit Learning in Autoencoder Novelty Assessment. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02*, volume 3, pages 2878–2883, 2002.

[69] V. N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

[70] V. N. Vapnik and A. Ya. Chervonenkis. *Theory of Pattern Recognition [in Russian].* Nauka, USSR, 1974.

[71] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. Software available from scipy.org.

[72] I. F. Weustink, E. ten Brinke, A. H. Streppel, and H. J. J. Kals. A generic framework for cost estimation and cost control in product design. *Journal of Materials Processing Technology*, 103(1):141–148, 2000.

Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Marcel Öfele — Magdeburg, 25th January 2021