

Zhihao Tan

PRODUCT SHOULD-COST ESTIMATIONS BASED ON WAREHOUSE DATA

Master's Thesis in Technology
Faculty of Information Technology and Communication Sciences
Examiner: Konstantinos Stefanidis
Examiner: Jyrki Nummenmaa
April 2024

ABSTRACT

Zhihao Tan: Product should-cost estimations based on warehouse data
Master of Science thesis
Tampere University
Master's Degree Programme in Computing Sciences
April 2024

Product should-cost estimation is essential for manufacturing industries since it brings a direct impact on organizational profitability and strategic sourcing. In practice, the should-cost of a product is calculated based on various cost analysis methods which may differ from the actual agreement price made between suppliers and purchasers. It possibly results in overpaying for the product and losing potential profits during the sourcing process. Therefore, for this thesis study, it is needed to develop a costing model that accurately estimates the should-cost of a product and visually displays the price difference between the estimated cost and the agreement price. Furthermore, as an outcome of this study, the created model would provide insights into the procurement process and give suggestions for making the best purchasing decision.

The concept of should-cost arose as a methodological approach for comparing the price between the quotation obtained from the supplier and the actual price a product should cost. There was relatively limited existing research about should-cost analysis and rare current studies offered clear suggestions about how should-cost estimation should be implemented. Comparatively, there were more appearances of commercial software solutions in the market. Therefore, the purpose of this study is to implement a general machine learning pipeline for exploring the should-cost estimation problem. Additionally, the experiments and analyses were applied to the selected product category to ascertain the current cost, and multiple regression models were utilized in order to find the linear relationship of the product-related known historical data for making cost estimation of possible similar new products in the future.

This study utilizes machine learning models, which include multivariate linear regression, K-nearest neighbors regression, support vector regression, lasso regression, ridge regression, decision tree regression, and random forest regression. Since no machine learning model can provide analyses for the cost across all the product categories, these selected models offer the benefits of selecting the most suitable approach. In this study, the estimations were made based on the characteristics of the particular product category. The original historical data was obtained from several data warehouses of the case company. The data was used in the created pipelines to make estimations and relevant discussions were made for the model results. The evaluation metrics used for evaluating the models in this study include mean squared error, mean absolute error, coefficient of determination, and cross-validation.

The result of this study indicated that the lasso, ridge, and random forest regression models performed the best results after validation, while the rest exhibited relatively weaker performances. This study aimed to make contributions from three perspectives. To begin with, the existing research and relevant literature were highlighted to satisfy the need to establish theoretical understandings of models and methods for product cost estimation. Later on, a comprehensive pipeline for data collection, preprocessing, and model training was realized based on the necessity of the study. Consequently, statistical interpretation of data and results were exhibited for understanding the analyses. Lastly, practical implementations were conducted in this study to contribute to the limited research of this topic, such as enhancing the domain of should-cost and supporting sourcing to assess the reasonableness of the quoted price from suppliers.

Keywords: product cost estimation, should-cost estimation, machine learning, regression model, data exploration, model evaluation, data manipulation, cross-validation

The originality of this thesis has been checked using the Turnitin Originality Check service.

PREFACE

The thesis study has been a long journey for me, from having very little knowledge and experience related to product cost estimation and machine learning model theories to completing and implementing the general processes. It has been many days and nights of reading and exploring, and the outcome is positive from my perspective.

For the thesis work, the idea was formed at the beginning of 2023, and the thesis lasted about a year. The general implementation was done earlier than the finalization of the thesis writing.

During the journey, I highly appreciated this thesis topic offered by Markku Mäenpää and Ilkka Ristiluoma. Thank you Markku for your guidance during the past few years and the thesis work support! Thank you Ilkka for raising the initial idea of the investigation and giving support for the specific product information as a study case!

I am so grateful for getting the support from Professor Konstantinos Stefanidis and Professor Jyrki Nummenmaa. Thank you Kostas for providing suggestions and sharing ideas about my thesis implementation and the documentation for experimental results! Thank you Jyrki for supporting my thesis writing and raising the points that needed to be corrected!

I also want to thank Juho-Pekka Koponen for your constant support in checking the implementation and giving improvement tips for the writing process!

Lastly, I want to thank Zixin He for your accompany and support in life!

Vaasa, 8th April 2024

Zhihao Tan

CONTENTS

1.INTRODUCTION.....	1
1.1 Research background.....	1
1.2 Research questions and objectives	2
1.3 Scope and delimitations.....	4
2.LITERATURE REVIEW.....	5
2.1 Overview of product cost estimation	5
2.2 Machine learning in cost estimation	8
2.3 Related work for cost estimation	9
2.3.1 Estimating manufacturing costs in mechanical engineering	10
2.3.2 Estimate product costs in the early design stage.....	12
2.3.3 Cost estimation with explainable artificial intelligence.....	14
2.3.4 Linear performance pricing to predict optimal price	15
3.METHODOLOGY.....	18
3.1 K-nearest neighbors regression	18
3.2 Linear regression.....	19
3.3 Support vector regression.....	22
3.4 LASSO regression.....	26
3.5 Ridge regression.....	27
3.6 Decision trees.....	28
3.7 Random forests	31
4.DATA PREPARATION.....	33
4.1 Data collection	33
4.2 Data cleaning.....	35
4.3 Feature engineering.....	37
4.3.1 Transforming numerical feature	37
4.3.2 Transforming categorical feature.....	39
4.4 Data exploration	40
4.4.1 Data distribution	40
4.4.2 Pairwise comparison.....	42
4.4.3 Outlier filtering.....	43
4.5 Feature Selection.....	47
5.MODEL TRAINING AND EVALUATION	50
5.1 Data splitting.....	50
5.2 Evaluation metrics	51
5.3 Model training.....	52
5.3.1 Exploring multivariate linear regression.....	53
5.3.2 Exploring K-nearest neighbors regression.....	55

5.3.3 Exploring support vector regression	56
5.3.4 Exploring lasso and ridge regression	57
5.3.5 Exploring decision tree and random forest regression.....	59
6.RESULTS AND DISCUSSION.....	62
6.1 Overview and comparison of model results	62
6.2 Challenges of current work	64
7.CONCLUSION	66
REFERENCES.....	68

LIST OF FIGURES

<i>Figure 1. Product cost estimation techniques [1].</i>	6
<i>Figure 2. Cost estimation methods evaluation at different phases [3].</i>	11
<i>Figure 3. Feature correlations of the collected dataset [3].</i>	11
<i>Figure 4. Machine learning pipeline for the cost prediction [11].</i>	13
<i>Figure 5. Research Framework [12].</i>	14
<i>Figure 6. Service scenarios [12].</i>	15
<i>Figure 7. Linear Performance Pricing [15].</i>	16
<i>Figure 8. Support vector machine trained with two sample classes. (a) Maximum-margin hyperplane and margins for an SVM. (b) Hyperplanes to separate the classes [25].</i>	22
<i>Figure 9. Hyperplanes in different dimensions [27].</i>	23
<i>Figure 10. A typical regression of SVR [26].</i>	24
<i>Figure 11. Sketch of simplified tree-growing algorithm [41].</i>	29
<i>Figure 12. Pipe clamp from engineering drawing. (a) Pipe clamp with one pipe-type. (b) Pipe clamp with two pipe-type.</i>	33
<i>Figure 13. The data-dense display created with Missingno.</i>	36
<i>Figure 14. Summary of normalization techniques [57].</i>	39
<i>Figure 15. The data distribution of numerical features.</i>	41
<i>Figure 16. The data distribution plot of WEIGHT_NETTO feature. (a) Before log normalization. (b) After log normalization.</i>	41
<i>Figure 17. Scatter plot between numerical features and the target variable.</i>	43
<i>Figure 18. A set of numerical features with boxplot (blue) and strip plot (yellow).</i>	44
<i>Figure 19. Identify outliers from the target variable. (a) A boxplot for the target variable. (b) A strip plot for the target variable was applied with 0.2 jitters to reduce overplotting.</i>	45
<i>Figure 20. Correlation matrix between features with 2 decimal places after outlier filtering</i>	47
<i>Figure 21. The designed pipeline to train with regression models</i>	53
<i>Figure 22. Exploring multivariate linear regression. (a) Feature importance. (b) Comparison between ground truth and prediction.</i>	54
<i>Figure 23. Exploring K-Nearest Neighbors regression. (a) Feature importance. (b) Comparison between ground truth and prediction.</i>	55
<i>Figure 24. Exploring support vector regression. (a) Feature importance. (b) Comparison between ground truth and prediction.</i>	57
<i>Figure 25. Exploring lasso regression. (a) Feature importance. (b) Comparison between ground truth and prediction.</i>	58
<i>Figure 26. Exploring ridge regression. (a) Feature importance. (b) Comparison between ground truth and prediction.</i>	58
<i>Figure 27. Exploring decision tree regression. (a) Feature importance. (b) Comparison between ground truth and prediction.</i>	60
<i>Figure 28. Exploring random forest regression. (a) Feature importance. (b) Comparison between ground truth and prediction.</i>	60
<i>Figure 29. The comparison of K-Nearest Neighbors regression model results before and after hyperparameter tuning.</i>	63
<i>Figure 30. The comparison of lasso regression model results before and after hyperparameter tuning.</i>	64
<i>Figure 31. A boxplot of labelled prices across different vendors.</i>	65

LIST OF TABLES

<i>Table 1. Variables table based on characteristics of target product category.</i>	<i>34</i>
<i>Table 2. Variables table based on class attributes of target product category.</i>	<i>35</i>
<i>Table 3. The pairwise correlation between features and the target variable.</i>	<i>45</i>
<i>Table 4. The correlation coefficients between PO, MAP, and AGR.</i>	<i>48</i>
<i>Table 5. Count of unique categories in a categorical feature.</i>	<i>49</i>
<i>Table 6. Evaluation results for used models.</i>	<i>62</i>

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
CAD	Computer-aided Design
CART	Classification and Regression Trees
CV	Cross-Validation
DT	Decision tree
Grad-CAM	Gradient-weight Class Activation Mapping
ID3	Iterative Dichotomiser 3
K-NN	K-nearest Neighbors
LASSO	Least Absolute Shrinkage and Selection Operator
LPP	Linear Performance Pricing
LSE	Least Squares Error
LSRT	Least Squares Regression Tree
MAE	Mean Absolute Error
ML	Machine learning
MSE	Mean Squared Error
OEM	Original Equipment Manufacturer
OLS	Ordinary Least Squares
PCE	Product Cost Estimation
RF	Random forest
RMSE	Root Mean Squared Error
RSS	Residual Sum of Squares
SVC	Support Vector Classification
SVM	Support Vector Machine
SVR	Support Vector Regression
XAI	Explainable Artificial Intelligence
b	Bias
β	Regression Coefficient Vector
$\hat{\beta}$	Least Squares Estimator
C	Regularization Parameter
ϵ_i	Random Component
L	Distance Between Support Vectors
R^2	Coefficient of Determination
x_i	Independent Variable
y_i	Dependent Variable
\bar{y}	Sample Mean
\hat{y}	Predicted Value
ω	Weighted Vector
ξ	Slack Variable
θ_k	Random Vector
μ	Mean
σ	Standard Deviation

1. INTRODUCTION

1.1 Research background

Product cost estimation (PCE) is an increasingly popular topic in manufacturing industries which takes a critical role in optimizing production performance and managing expenses. On the one hand, overestimating the product cost may lead to the loss of business opportunities; on the other hand, underestimating the cost will result in financial losses [1]. It can be said that PCE is closely related to financial efficiency and strategic sourcing, especially during the manufacturing and procurement processes.

The concept of should-cost analysis, raised by global management consulting company McKinsey [7], is to determine the optimal cost that a product should have. This analysis becomes especially important in the situation when agreement prices are made with suppliers. It usually happens to have a distance between the agreement price and what the product is supposed to cost. It functions as a benchmark for evaluating supplier quotations and plays an important role during the strategic sourcing and procurement negotiation processes. The data-supported estimation approach can be realized for predicting the should-cost of a new product by utilizing historical data of the same product category. As a result, the approach can provide suggestions for reasonable costing, and therefore to some extent ensure the positive outcome of negotiation and profitability.

This thesis study applied machine learning (ML) methods to analyse the historical data collected from several data warehouses from the case company. The feature engineering methods were applied to the data to create the regression models. The target is to make should-cost estimations of an existing product or a new product with various key cost drivers. Several ML models were selected for making the product should-cost estimation, for instance, multivariate linear regression (MLR) and random forest (RF). Through the process of analysing the historical data, the created ML models are trained to be able to estimate what a product should cost. The trained model can be used as a tool to provide support for purchasers to get cost-saving recommendations

when they negotiate the product or service price with suppliers. Furthermore, the estimation result helps to align the predicted price and the agreed price of the specific product by referring to historical records and related cost factors. However, the analysis process of should-cost can be time-consuming and complicated since it has the prerequisite of the domain-related experience of experts [43]. Therefore, the purpose of this study is to use ML techniques with the combination of expert judgement to make the should-cost estimation efficient and accurate.

1.2 Research questions and objectives

In the manufacturing PCE field, expert judgement and historical cost data analysis methods are one of the most frequent-applied approaches for determining what a product should cost. These methods are established on the basis of the extensive personal experience of experts and their product-related knowledge from historical projects. Therefore, these methods are expected to provide valuable understandings of cost estimation processes during different periods of a product lifecycle [2].

Since the PCE knowledge is complicated and the sources are various, there is an increasing recognition of the need to apply advanced technologies to current methods. The expert judgement and historical data analysis are known as typical qualitative approaches [1]. The combination of these two methods provides a general understanding of cost drivers [2]. The integration of the qualitative-based methods with the quantitative-based ML techniques is called the mixed approach. It combines the advantages of expert judgement method, historical data analysis, and ML techniques, and as a result, a more comprehensive and accurate should-cost estimation can be realized.

This mixed approach is regarded to be more convincing when it is applied to products with similar product categories. The reason is that no estimation model can satisfy all the scenarios in the PCE process. Thus, gathering product data with the same product category and focusing on the characteristics of the filtered product group can make a greater impact on the estimation results.

The research questions of this study include:

- (1) What results does combining ML models with expert judgement have on the accuracy of should-cost estimations for specific product categories?
- (2) What are the results of using trained regression models to estimate product should-cost in strategic purchasing between the purchasers and suppliers?
- (3) Are the quoted prices from suppliers consistent with what products or services should cost?

To answer the first research question, the following steps were taken. Firstly, the theories based on the selected models were studied. Secondly, the costing experts were contacted to identify the cost drivers that are most correlated to the product. Thirdly, the models were trained with collected data for comparing the estimated prices and actual prices. Lastly, the realizability and results of this mixed approach were determined.

To answer the second and third research questions, this study has applied the historical data of a specific product category to train the models. The trained models were used for making estimations for the new product when a new product comes with needed attributes. This is because products in the same category are usually made in a similar way. The estimation provided an initial price for the new product based on similar previous manufactured products. As a result, this supports purchasers to make judgements of the quoted product cost from suppliers and identify whether the price is in line with its actual value.

The main objective of this study is to figure out the ML model that performs the best result for the product should-cost estimation problem. The selected ML models are applied to the collected datasets for the comparison of the predicted values with the actual product prices. The result of this thesis study shows the trained models with the highest performance and accuracy for supporting price negotiations. Furthermore, this study provides practical implications for the topic and makes recommendations for similar research in the future.

1.3 Scope and delimitations

As mentioned earlier, it is unrealistic to have a prediction model that fits all the product categories. By analysing the characteristics of a specific product group through data exploration, the created prediction model can achieve higher accuracy and relevance. Because products within the same category often share a similar manufacturing process, drawing, and cost structures. Furthermore, similar category items are likely to be influenced by a common set of factors, for instance, weight and raw material. These factors are expected to have similar impacts across products in the same group.

This specific product category approach focuses on the common factors and learns from historical data within this narrower scope. Then the trained ML model can be tuned to make more precise predictions and be more adaptable to actual cost drivers.

Hence for this study, the specific pipe clamp product group is chosen as a baseline. The analysis starts with pipe clamp historical data for similar category items and uses the data to make should-cost estimations of existing items and future new items. The same approach will be further practiced and examined with the product categories such as bended pipe and metal sheet.

For this thesis study, the predictive models are trained by utilizing collected historical data. It includes various factors that may influence the final product cost. The factors include material-related characteristics, design groups, class attributes from drawing, and more potential features. However, factors such as raw material trends, labour cost, technical values (for instance, bending amount, welding amount, thickness), and general manufactural process are excluded from this thesis analysis because those data are not available at the moment.

2. LITERATURE REVIEW

To provide solid support for conducting the thesis study, relevant theories are reviewed and applied. This chapter clarifies the key information of each knowledge point. In the below sub-chapters, Section 2.1 demonstrates an overview of PCE and should-cost, Section 2.2 illustrates ML in PCE, and Section 2.3 introduces previously achieved cost estimation.

2.1 Overview of product cost estimation

In the international market, the high intensity of global competition makes manufacturers compete based on product quality, product cost, and market time. Facing the uncertainty of the organizational and market environment, this study case needs to consider fairness pricing of the components. This means that understanding the cost of production, especially the cost of machined parts is crucial for efficient operation and competitive production [4], which is also the key discussion point of this study.

Product cost estimation (PCE) is identified as a key concept of this study. It is critical for manufacturing since it supports decision-making in the design and development phase as well as production and procurement. The reason is that PCE provides the fundamentals for pricing strategies and budget planning. In addition, it functions as a benchmark for evaluating supplier quotations and plays an important role during the strategic sourcing and procurement negotiation processes.

One of the existing research claimed that PCE methods have been categorized and divided into specific sub-methods: (1) intuitive methods, which rely on the experience of estimators; (2) analogical methods, which compute the cost of a product to its similar one; (3) parametric methods, which use design parameters and equations, and (4) analytical methods, which evaluate the cost by breaking down the products into cost elements [4]. This thesis puts more emphasis on the first and second methods.

Based on the previous work conducted by Ben-Arieh & Qian, further progress and contributions were made by Niazi et al. to this research field. In their research, a comprehensive review of PCE techniques was provided with an extensive hierarchical technique classification [1]. Furthermore, they categorized the PCE techniques into qualitative and quantitative techniques, which are explained in Figure 1.

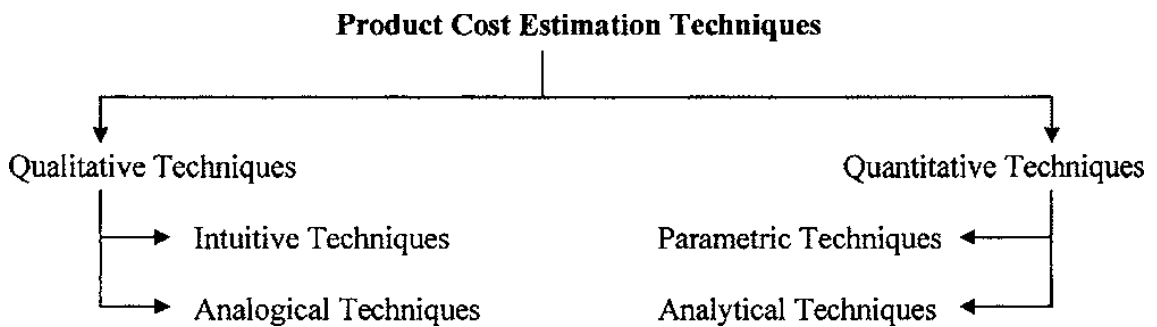


Figure 1. Product cost estimation techniques [1].

Both quantitative and qualitative techniques are frequently applied in PCE processes. One of the main differences is that they are applied at the different stages of product development. The former is used more often at the middle and late development stages to provide precise cost estimations, while the latter is already suited for an early development stage [3]. Below are definitions and explanations of these two methods.

The qualitative PCE techniques in manufacturing involve activities of comparing a new product with previously manufactured ones to identify similarities. Aiming at making cost estimations for the new products with similar past designs of it, the comparison activities involve several factors. For example, past design, manufacturing data, possible known costs of the product, and estimators' experience. It can be summarized that qualitative methods provide initial rough cost estimates during the early design stage and can be classified into intuitive techniques and analogical techniques [5] [1].

The quantitative PCE methods differ significantly from the qualitative methods, in terms of the stages when applying the methods. The quantitative methods put more emphasis on the detailed analysis of the product's specific characteristics, design, features, and manufacturing process [1] [3]. By using quantitative methods, the costs are calculated with various product-related parameters to the analytical function or the accumulation of the costs of each resource utilized throughout the entire production cycle.

It is regarded that the accuracy of quantitative methods for PCE is typically higher than qualitative approaches because it has more reliance on detailed product information. Quantitative methods are mainly applied to the final stage of the product design cycle due to two aspects. On the one hand, this approach has the requirement for detailed design information [5]. On the other hand, it requires understanding the design and features of the final product, which is often not able to be attained in the early development stage [3]. Furthermore, for the types of quantitative PCE methods can be further divided into parametric and analytical techniques.

The term should-cost applied in PCE, is another critical concept of this thesis. It refers to the ideal or optimal cost that a product, component, or service should have [7]. It is an estimation that combines cost calculation theories and the experience of the experts. One of the main targets of this study is to give suggestions for strategic sourcing and price negotiations, which can be accomplished by applying should-cost analysis. This analysis is a critical topic in cost estimation because it helps to evaluate and judge the reasonableness of the supplier quotations and it acts as a baseline for making internal goals and budgets. However, there is a limited amount of research that explains the general approach of how should-cost relating to PCE works and provides an empirical example. Furthermore, should-cost analysis is different from traditional cost estimation methods in terms of its emphasis point. While the traditional ones focus on determining the product cost based on historical data and current price, the should-cost analysis emphasizes defining what the cost should be. The should-cost analysis is selected for this study. Because the process of this type of method includes the comprehensive analysis of the manufacturing process, the raw material price, the labour market price, and other cost drivers.

The target of this study is to predict the should-cost of a product and select the key cost drivers based on a similar group of items. These items are selected with the consideration of possible known cost and material-related characteristics data. The factors such as labour, raw materials cost, season influence, location, and other things will not be considered because these are not currently available. It can be summarized that the analyses are conducted by using existing data and considering the previously set scope of analyses. The more complicated factors may be verified through Rapidminer software as planned.

2.2 Machine learning in cost estimation

Machine learning (ML), a subset of artificial intelligence (AI), is the process of training a model with massive amounts of data sets. It can create a mathematical relationship between the training data and the label to make useful predictions. ML can be classified into supervised learning, unsupervised learning, reinforcement learning, and generative AI based on how they make predictions and generate content. [8]

Supervised learning uses labelled data to map specific features to labels. It trains with known correct output to make the prediction and then find the relationships between the data and the correct known results. It can be applied to two primary cases, which are regression and classification. The regression problems predict numerical values, whereas the classification problems predict discrete outcomes to categories. Unsupervised learning makes predictions based on unlabelled data without known correct answers. It uses clustering to categorize related examples (one row of a dataset is an example) into groups based on features but no label [9] [10].

Reinforcement learning involves getting rewards and penalties for a training task through a loop until it finds out a strategy that gets the most rewards. Generative AI focuses on creating content from user input, it can generate text, images, or music as well as give a summary for provided articles or other human-created content [8].

As a critical method for conducting cost estimation and product development, ML is well known and claimed to be increasingly popular in this field. ML is applied for this thesis study since it achieves the necessity to bring historical data into the statistical algorithms for predicting the costs of new products. Analysing past manufacturing data enables companies to develop predictive ML models and make it possible to adjust and improve over time. As a consequence, these models can handle large data sets with multiple input features and identify patterns and trends. Subsequently, the relationships between input features and possible known costs can be discovered.

Some literature argues about how to develop a successful ML model for cost estimations. It is regarded that success is highly dependent on the quality and relevance of the selected input features. In order to be consistent with the literature, this thesis follows the general key steps: (1) Gathering relevant data; (2) Preprocessing the data for cleaning and transforming; (3) Splitting the data into training, testing, and validation sets; (4) Training the model and adjusting the weights of attributes; (5) Evaluating the accuracy of the model; and (6) Deploying the final model [11] [13].

There are plenty of discussion about utilizing ML methods for cost estimation purposes in existing research. For instance, ML models (such as regression model and RF model) can be used in automotive industries to predict the cost of automotive parts based on input features (for instance, material cost, size, and production volume) in the early product design phase [11]. Moreover, ML models can also be used in manufacturing industries to predict the cost by analysing 3D computer-aided models with a 3D deep learning architecture of machining features of parts [12].

In conclusion, ML techniques have the capability of analysing historical data and modelling complex problems which hardly can be programmed to provide accurate estimated values. Therefore, using ML techniques can significantly enhance cost estimation in manufacturing.

2.3 Related work for cost estimation

In the following sub-chapters, Section 2.3.1 presents a study about PCE in mechanical engineering, Section 2.3.2 discussed about PCE in the early design stage of the development, Section 2.3.3 illustrates a practical approach to make cost estimation by using explainable artificial intelligence, and Section 2.3.4 shows a case study of cost estimations with a linear performance pricing model.

2.3.1 Estimating manufacturing costs in mechanical engineering

Christoph et al. [3] present a novel ML-based method to estimate the manufacturing costs in the field of mechanical engineering. It shows precise predictions made in the early stage of the product development life cycle are possible by using little-known information about the final product. However, cost estimation is often associated with high uncertainty because of the limited information related to the final product at the early development stage. Furthermore, overestimation and underestimation can directly influence the company's performance and benefits [1].

This study evaluates extensive cost estimation methods and illustrates that quantitative methods often need an adequate understanding of product-specific characteristics. The quantitative methods are typically applied during the middle to late stages of product development. These are often based on data that is related to previous development and are generally utilized during early and middle product development stages. This is because most product-related features and advanced designs are gradually finalized at the middle and late stages of development, thus the estimations need to be made based on previous similar developments in the early stages of development. The evaluation results are shown in Figure 2 by utilizing a “+” symbol to indicate the method is suitable, a “-” symbol to denote the method is not suitable, and a “o” symbol to represent applicability under specific conditions.

Cost Estimation Methods			Early Phase	Mid Phase	Late Phase	
Quantitative Methods	Parametric Methods	Kilo Cost Method	0	-	-	
		Material Cost Method	0	+	-	
		Design Equations	-	-	+	
		Cost Functions	-	-	+	
	Analytical Methods	Operation-based Cost Models	-	-	+	
		Break-down Cost Models	-	-	+	
		Cost Tolerance Models	0	0	0	
		Feature-based Cost Models	-	0	+	
Activity-based Cost Models	-	0	+			
Qualitative Methods	Intuitive Methods	Case-based Systems	+	0	-	
		Decision Support Systems	Rule-based Systems	0	0	-
			Fuzzy Logic Systems	+	0	-
			Expert Systems	+	+	+
	Analogue Methods	Regression Analysis Model	-	+	0	
SW	/	Back Propagation Neural Network Model	-	+	0	
		Constructive Cost Model	0	-	0	
		Constructive System Engineering Cost Model	0	0	0	

Figure 2. Cost estimation methods evaluation at different phases [3].

This study utilizes a generic dataset comprising various mechanical domain-specific features such as length, weight, machine type, power supply, energy consumption, and performance (can be seen in Figure 3). Then the datasets were processed with different selected predictive ML algorithms such as linear regression, decision tree (DT), RF, K-nearest neighbors (K-NN), and support vector regression (SVR). Furthermore, the evaluation of predicted results has been done through root mean squared error (RMSE) and mean absolute error (MAE).

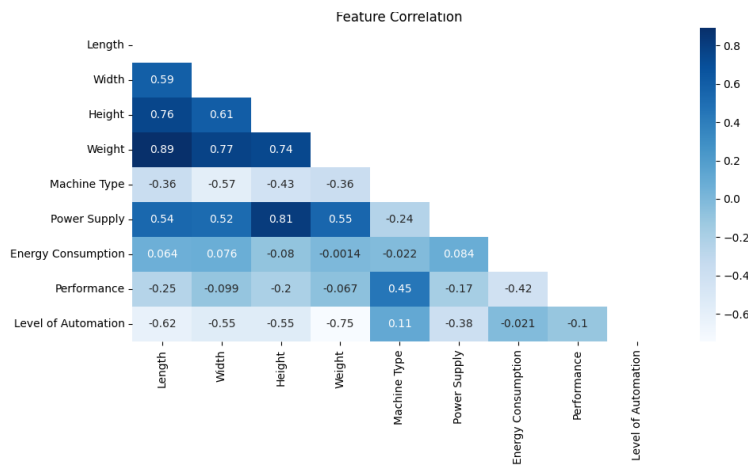


Figure 3. Feature correlations of the collected dataset [3].

The results indicate that K-NN and linear regression algorithms performed the best with prediction accuracy. However, the size and balance of the training data set for an algorithm to learn is a problem in estimating the costs.

2.3.2 Estimate product costs in the early design stage

Frank and Jörg [11] conducted a comprehensive study regarding estimating accurate product costs at the early design stage of the development. The study applied different potential ML techniques with the real-world data collected from an Original Equipment Manufacturer (OEM).

The collected data used in this study is saved at OEM by using “Siemens Teamcenter Product Cost Management” software. The data contains wheel cost data of 1340 automobiles. The researchers also mentioned what data to be explored in the ML model, which includes basic data (such as product lifetime, and location), material-related data (for instance, quantity, weight, and raw material price), manufacturing steps, assembly steps, process steps and so on.

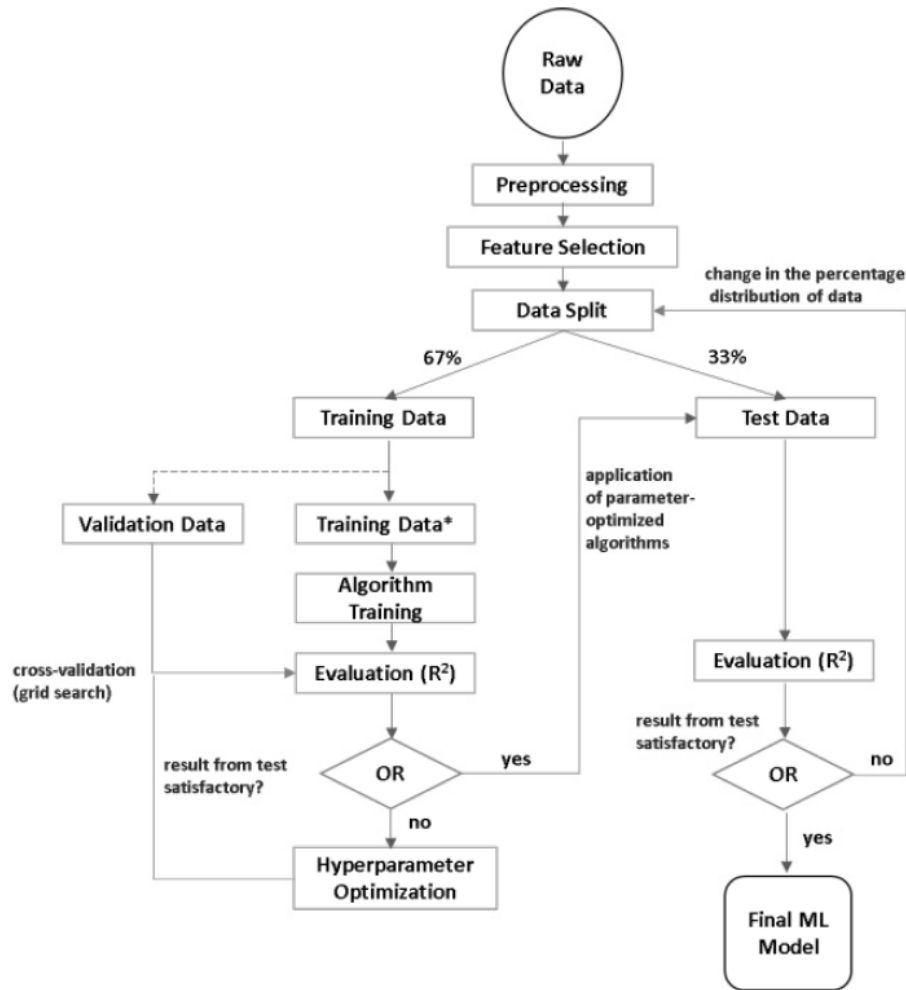


Figure 4. Machine learning pipeline for the cost prediction [11].

The designed ML pipeline in this study can be seen in Figure 4, which includes data collection, data preprocessing, feature engineering, train-test data split, and evaluation. The model used in this study includes linear regression, SVR, DT, ensemble, artificial neural networks, and K-NN. The performance metrics of used algorithms from this study show a high R^2 score, which indicates all the involved ML algorithms make estimations with high accuracy. In addition, the study points out that the challenge of PCE is to get good quality and quantity of training data, and a more consistent database can comprehensively improve the predictive power of the model.

2.3.3 Cost estimation with explainable artificial intelligence

The study conducted by Soyoung and Namwoo [12] experienced a deep learning-based explainable AI (XAI) approach to make manufacturing PCE process for known 3D computer-aided design (CAD) models. In addition, they tried to visualize the machining features from the model to understand which feature has the most critical impact on manufacturing cost. As the designer might not have domain-specific knowledge directly to the cost estimation, the aim of this study is to inform the designer which part of the CAD drawing needs to be modified to reduce the manufacturing cost.

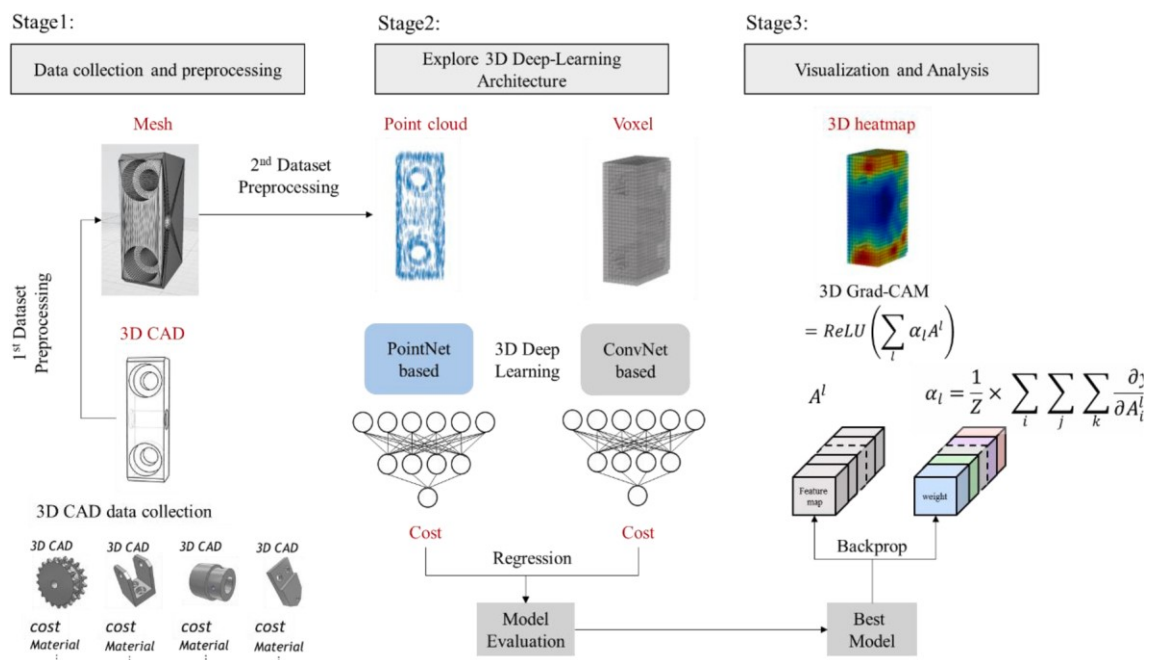


Figure 5. Research Framework [12].

The proposed process from this study is shown in Figure 5 which contains:

- 1) Data collection and preprocessing: from which the machining parts data such as 3D CAD drawing, cost, volume, and material were gathered and pre-processed to be used as input to the deep learning model.
- 2) Exploration for 3D deep learning architecture: from which the various cost prediction architectures have been evaluated and the best model will be selected after the evaluation.

- 3) Visualization and explanation for the result: from which the 3D gradient-weight class activation mapping (Grad-CAM) is executed to interpret and explain the result of predicted cost.

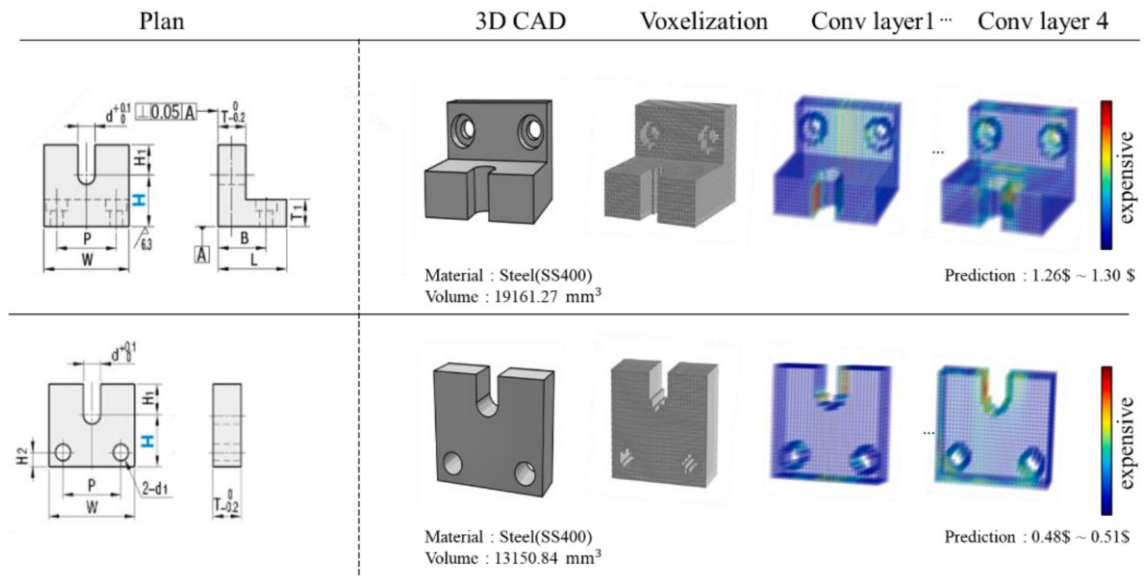


Figure 6. Service scenarios [12].

A use case of this study can be seen in Figure 6, which shows that designers can input 3D CAD drawings into the created model that automatically transfers the drawing into voxels. Then the model finds areas that impact the manufacturing cost visually so that designers know which part potentially affected the cost.

2.3.4 Linear performance pricing to predict optimal price

Tanak and Yilmaz [14] showed a case study that used the linear performance pricing (LPP) method to determine the correct price for a product or part, and finally beneficial to procurement saving, purchasing, and price processing by altering the product cost to its optimal price. This study mentioned the cost-saving strategy LPP model, which was originally developed and introduced by consulting company McKinsey. The model allows companies to negotiate the reduced cost of an identified material (parts belonging to the same type of material group) as well as helps suppliers adjust the price and pro-

cess. A similar study also shows that LPP is beneficial for negotiating the cost for currently purchased or newly redesigned parts by utilizing a series of regression analyses with measured performance metrics [15]. The performance coefficients represent the contribution margin of each performance index (for instance, weight, quantity, diameter) to the price of the product.

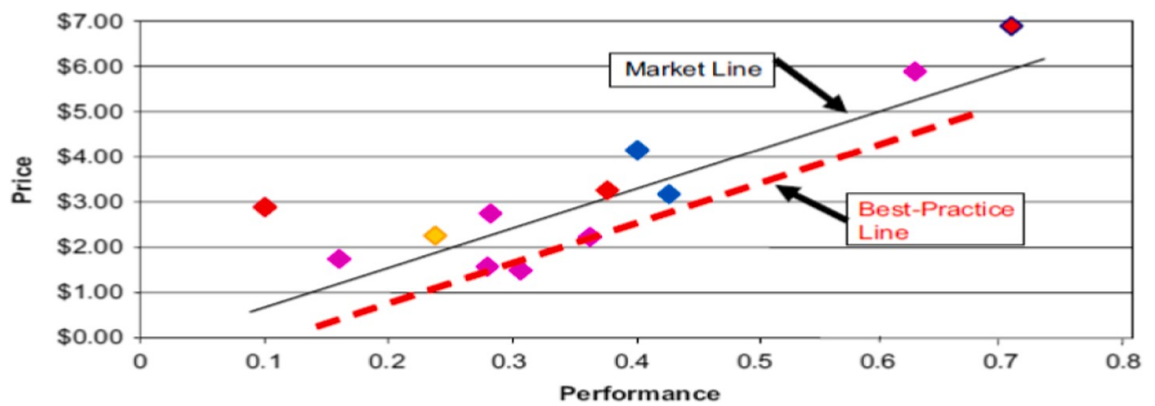


Figure 7. Linear Performance Pricing [15].

In Figure 7, it demonstrates the general LPP structure. For instance, consider an available product with a variety of models from vendors. This happens under the circumstance of the plotted graph about the price of each model versus performance metrics (key drivers like diameter, complexity, and quantity). At this moment, a linear relationship comes up with some model's prices above and below the line. The red dashed line is demonstrated as the best-practice line and the black solid straight line is shown as the market line [15].

The dataset applied in this study was gathered from the purchasing department of an OEM company to determine the optimal price for connection materials (fittings). The main criteria that affect the price are determined by the purchasing expert of the company, for instance, the key cost drivers used for this study include weight, quantity (annual consumption amount), and diameter. This study mentioned that they first used the LPP method to determine fasteners from certain group out of in total 1000 types of fasteners. Then the optimal cost is determined for the products that belong to the same group and verified with fuzzy logic.

The result from this study shows that the LPP is used in the supply chain network and other fields by sourcing or purchaser to analyse the relationship between vendor and OEM, and the relationship between performance metrics of product and cost. It is beneficial for cost saving, price negotiation, and determining the best price.

3. METHODOLOGY

This chapter discussed the selected regression models based on previous literature. The thesis aims to estimate the should-cost of a product with machine learning models. Thus, the most common approaches are applied in the thesis. The models are introduced in each section of this chapter sequentially, which include K-nearest neighbors regression, linear regression, support vector regression, lasso regression, ridge regression, decision tree regression, and random forest regression.

3.1 K-nearest neighbors regression

The K-Nearest Neighbors (K-NN) is known as a supervised ML algorithm, and it is commonly applied in classification and regression tasks. The K-NN is selected for this study since there is a popularity to apply this method in manufacturing industries and machine engineering [16]. When applying the K-NN algorithm to classification tasks, the main aim is to get the label that is classified from its neighbors through a voting technique. In this situation, the voting technique selects the most frequent label from its nearest neighbors as the output. While utilizing K-NN for regression problems, the label is typically found out by the average value of its nearest observations [17].

For this study, the purpose is to use the ML models in regression problems, thus K-NN is applied with the target of predicting the cost of a product. The steps for K-NN regression can be demonstrated as follows:

- Computing the distance from the observations to the target data point.
- Sorting the computed distance with an ascending order
- Finding an optimal K value with cross-validation (CV) and learning curve.
- Returning the average value of K nearest observation as a prediction

The distances between the observations and target data point are calculated by (1) choosing a proper similarity measurement to compute the distance (such as cosine similarity), or (2) selecting a suitable distance metric (such as Euclidean distance and

Manhattan distance). The Euclidean distance metric is the most frequently used distance function. The general form of the function in n -dimensional Euclidean space can be written as [18] [19]:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

where x and y are two points of cartesian coordinates, x_1, x_2, \dots, x_n are the record of variables for the first point, and y_1, y_2, \dots, y_n are the record of variables for the second point.

Concerning the interpretation of hyperparameter K in K -NN models, it refers to the number of the nearest neighbors of the specific data point. For a specific ML problem, the choice of K represents the trade-off between overfitting and underfitting of the models. Moreover, the optimal value of K can be found by using CV [19]. The meaning of the value can be understood in the following ways. A small K value may potentially lead to overfitting because the target point closely follows the training data. A large K value can result in underfitting because the model cannot learn the training data correctly [20]. In addition, it may also be possible that a smaller value of K tends to result in high variance but low bias, while a larger value of K may lead to lower variance but high bias [21].

3.2 Linear regression

Linear regression (LR) is a fundamental statistical ML model, which tries to estimate a linear relationship between (one or more) independent and dependent variables under supervised learning [22]. The case with only one independent variable is defined as univariate linear regression; for the case of more than one independent variable is named as multivariate linear regression (MLR).

In the univariate linear regression model, the analysis takes an independent variable (regressor) to predict the dependent variable (response). The model equation can be explained as:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad (2)$$

where x_i refers to the independent variable, y_i refers to the dependent variable, and a collection of datasets can be denoted as (x_i, y_i) . The β_0 is the intercept, β_1 is known as the slope or the coefficient term, and ϵ_i refers to the unknown random component to handle residues. In addition, the residue is the difference between the estimated values and actual observed values.

Suppose the value of independent variable x_i is fixed, the random component ϵ_i of the defined univariate linear regression function indicates the property of dependent variable y_i [23].

In the MLR model, more than one independent variables are involved to form the model. The unknown coefficients $\beta_0, \beta_1, \dots, \beta_k$ imply the linear relationship of the model and the equation for k regressor variables can be written as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \epsilon \quad (3)$$

The MLR commonly be rewritten into a matrix notation [23]:

$$y = X\beta + \epsilon \quad (4)$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \quad \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \quad (5)$$

where y is an $n \times 1$ vector that represents the observation or dependent variable, X is an $n \times k$ matrix that contains all the inputs for the model, β is a $k \times 1$ vector that represents for model parameter or regression coefficient, and an $n \times 1$ vector ϵ which represents for random errors vector. In addition, the value of n indicates the number of observations or sample size, and k indicates the number of regressor variables.

Notation $\hat{\beta}$ is used to represent the estimator for β , and it estimates all the parameters in the model. The \hat{y} represents the predicted value, and the model residue ϵ equals the observed value y minus the predicted value \hat{y} .

$$\epsilon = y - \hat{y} \quad (6)$$

Hence, the equation can be rewritten as:

$$\hat{y} = X\hat{\beta} \quad (7)$$

The least squares estimator aims to minimise the square of residual ϵ and it is often used to find out the intercept β . Since minimizing the sum of squares of residuals from i equals 1 to n , is the same as minimizing the transpose of ϵ times ϵ . Therefore, the transpose of ϵ would be a $1 \times n$ vector and the equation of the sum of squares can be written as:

$$\sum_{i=1}^n \epsilon_i^2 = \epsilon'\epsilon = (y - \hat{y})'(y - \hat{y}) = (y - X\hat{\beta})'(y - X\hat{\beta}) \quad (8)$$

The derivative of the function is taken regard to the $\hat{\beta}$ and the value is set to 0 in order to find out the $\hat{\beta}$:

$$\frac{\partial}{\partial \hat{\beta}} (y - X\hat{\beta})'(y - X\hat{\beta}) = 0 \quad (9)$$

As a result, the least squares estimator $\hat{\beta}$ can be simplified as:

$$\hat{\beta} = (X'X)^{-1}X'y \quad (10)$$

where $(X'X)^{-1}$ is the inverse matrix of $X'X$ and X' is the transpose matrix of X [24].

3.3 Support vector regression

Support Vector machine (SVM) is one of the supervised algorithms that can be used for both classification and regression problems. The SVM can perform linear or non-linear based on selected kernel functions [28].

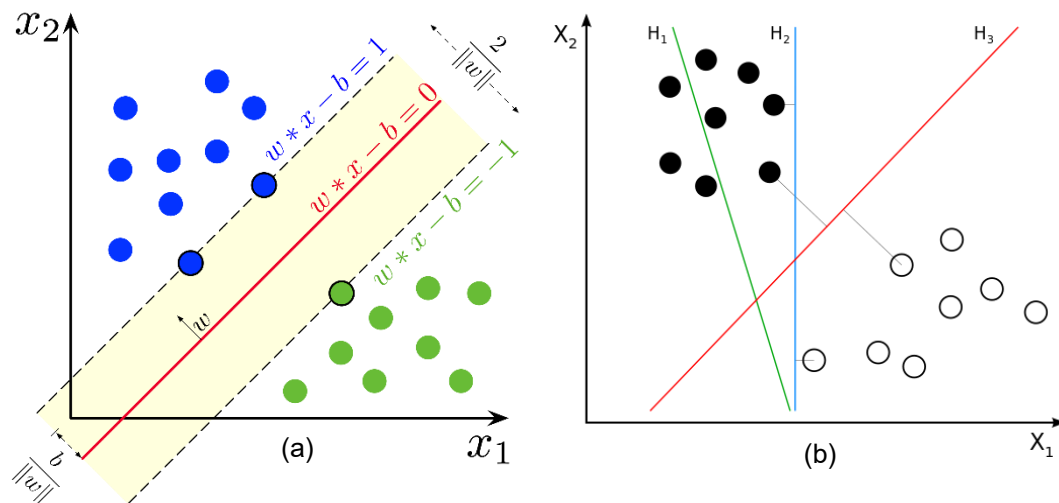


Figure 8. Support vector machine trained with two sample classes. (a) Maximum-margin hyperplane and margins for an SVM. (b) Hyperplanes to separate the classes [25].

In support vector classification (SVC) problems, the purpose is to find a hyperplane that gives the largest margin geometrically by maximizing the perpendicular distance of the support vectors to optimize the classification. The above figures explain the concept of SVC, which can be understood in the following way. In Figure 8 (a), there are two features x_1 and x_2 , and the target variable consists of two classes represented with blue and green circles. The example hyperplane splits the sample points into two parts

properly with the maximum margins. When it comes to Figure 8 (b), there are example lines H1, H2, and H3. The line H3 is considered the best hyperplane since it can give the maximum geometric distance with the support vectors. In addition, the sample points located on the margin (points on the dashed line) are called support vectors that indicate the structure of the SVM.

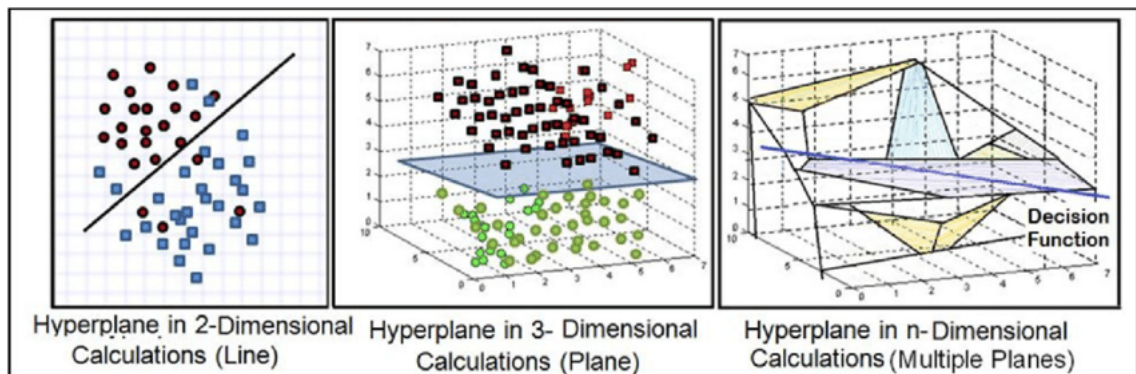


Figure 9. Hyperplanes in different dimensions [27].

The hyperplane discussed above is a subspace that has one less dimension than its ambient space. Figure 9 demonstrates how a hyperplane separates a space with n dimensions into two sides. There are different hyperplane understandings when considering different dimensions. When the space is regarded as a two-dimensional ambient space, any lines work as the hyperplanes for dividing the ambient space into two separate parts. Similarly, when the space is considered as a three-dimensional ambient space, any two-dimensional space functions as the hyperplanes for separating the ambient space into two identical parts. By following this logic, when the space is defined as a n -dimensional ambient space, any $(n-1)$ -dimensional space performs as the hyperplanes for splitting the ambient space into two different parts. Moreover, a single point is a hyperplane when the ambient space is a 1-dimensional line. However, the hyperplane that separates the ambient space into two may not be possible to find in a given dimension space. Therefore, kernel tricks are commonly used in SVM to increase the dimension of the ambient space to find a suitable hyperplane.

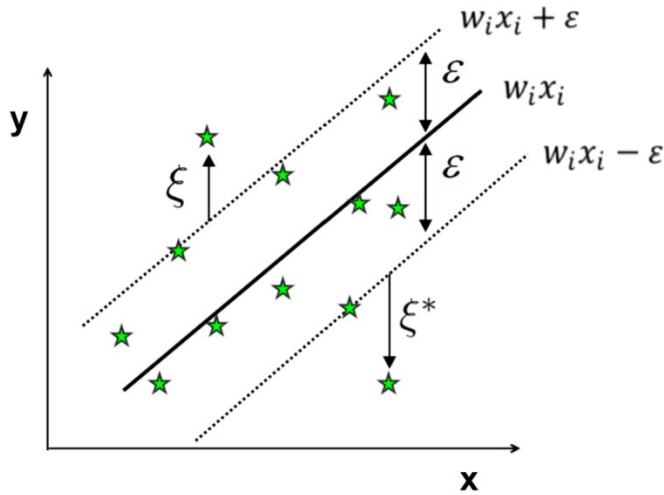


Figure 10. A typical regression of SVR [26].

In support vector regression (SVR), the aim is to find the best-fit flat hyperplane so that the majority of the training samples are within the margins. The role of margin in a regression problem behaves differently than it behaves in a classification problem. It functions as a decision boundary, and it is defined by the value of epsilon (epsilon indicates the tolerance level). As shown in Figure 10, the maximized training samples located inside the upper and lower boundaries will be avoided from outlier inclusion.

The SVR can use different types of linear and non-linear kernel functions to transform input vectors into the required type of processing data [28]. Moreover, the linear, polynomial, and Gaussian are commonly used functions in kernel tricks. For this study, SVR with linear kernel is used which is a dot product between two input vectors [30]:

$$y = \langle \omega, x \rangle + b, \omega \in \mathbb{R}^n, b \in \mathbb{R} \quad (11)$$

where the $\langle \omega, x \rangle$ illustrates the dot product, the ω means the weighted vector, constant b is an unknown bias, and a real coordinate space of dimension n is denoted as \mathbb{R}^n .

The distance L between the support vectors is written as:

$$L = \frac{2}{\|\vec{\omega}\|} \quad (12)$$

To maximize distance L between margin, which is equivalent to minimize a monotone function of weight:

$$\frac{1}{2} \|\omega\|^2 \quad (13)$$

The constraints show that all the sample points must located within the margin:

$$\begin{cases} y_i - \langle \omega, x_i \rangle - b \leq \varepsilon \\ \langle \omega, x_i \rangle + b - y_i \leq \varepsilon \end{cases} \quad (14)$$

where y_i means observations either above or below the prediction line, the predicted value is $\langle \omega, x_i \rangle + b$, and the margin is defined by ε .

The existing constraints are supposed to be defined to allow all the sample points to be located inside the margin. However, it may not be possible to satisfy the conditions for all the training sample points. As a solution, the slack variable ξ is applied to deal with these unrealizable cases in an optimization problem [29]. The sample points outside the margin are called slack variables, which gives the error allowance of the regression error to exist up to the value of ξ . After utilizing the slack variables ξ_i and ξ_i^* to the optimization problem, the problem changed to minimize the weight append with the summation of all the slacks:

$$\frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \quad (15)$$

The constraints are slightly changed after taking consideration of slack variables:

$$\begin{cases} y_i - \langle \omega, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle \omega, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \quad (16)$$

The hyperparameter C is a regularization constant, which balances the trade-off between flatness and the amount of tolerated deviation [29, 30]. For instance, giving a larger value to the tuneable parameter C makes the model not allow slack variables or the slack variables should be smaller in the minimization problem.

3.4 LASSO regression

LASSO (Least Absolute Shrinkage and Selection Operator) regression is a regularized linear regression method extended from ordinary least squares (OLS), which added an L1 penalty to the residual sum of squares (RSS) [31]. It is commonly used for feature selection because it shrinks parameters associated with less important features to zero. Thereby, it removes features that are not important in explaining the target variable.

The RSS tries to sum the squares of the residual difference between the actual value y_i and the predicted value:

$$RSS = \sum_{i=1}^n \left(y_i - (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \dots + \beta_j x_{i,j}) \right)^2 \quad (17)$$

The coefficient estimator $\hat{\beta}^{lasso}$ in lasso can be calculated by the minimization of RSS with the constraint condition [32]:

$$\hat{\beta}^{lasso} = \operatorname{argmin}_{\beta} \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p x_{i,j} \beta_j \right)^2 \quad \text{subject to} \quad \sum_{j=1}^p |\beta_j| \leq t \quad (18)$$

where the constant t defines the regularization, and it should be chosen adaptively to minimize the estimate of the expected prediction error [33].

By utilizing this approach to find the β_j that minimizes RSS, the more features used in the model, the smaller the training error will behave. However, it will relatively increase the complexity of the model. To solve this problem, the absolute magnitude value of the β_j is appended after the original formula as a penalty term (L1 regularization):

$$\hat{\beta}^{lasso} = \operatorname{argmin}_{\beta} \left(\frac{1}{2} \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p x_{i,j} \beta_j \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \right) \quad (19)$$

The newly added term to the equation gives a penalty for the model, meaning a large number of variables can make the sum of values of the β result in a higher penalty. In addition, a two is sometimes added to the denominator in the front for a computational reason and mathematical convenience, as it will be cancelled out after the derivative.

Then it comes to the interpretation of the tuning parameter λ in Equation 19. The value of it indicates the amount of shrinkage. It is usually estimated by CV with a range of possible options to find the optimal parameter [31] [33]. The tuning parameter λ is taking an important role as an influential factor in the result. If λ is small and reaches zero, then the lasso regression will be identical to OLS, and no penalty is added to the model. Similarly, a higher λ can make the penalty term get bigger and increase the bias.

3.5 Ridge regression

The equation of ridge regression is similar to lasso regression, the only difference is that the L1 lasso penalty is replaced by the L2 (the squared magnitude of the β_j) ridge penalty in ridge regression [33]:

$$\hat{\beta}^{ridge} = \operatorname{argmin}_{\beta} \left(\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p x_{i,j} \beta_j \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right) \quad (20)$$

In addition to the difference that appeared from the equation perspective, they also behave differently in terms of how they process input features for the model training. Lasso regression tends to exclude useless features, while ridge regression keeps all features even if some are irrelevant. In the situation when useless features can be removed from the model, the lasso regression tends to be a relatively good option. However, if there is a group of good features that are highly correlated, lasso regression tends to select only one of the features from the group. In this case, ridge regression can work as a better alternative since it keeps all the features in the model. This can be further proved in the later Chapter 5.3.4 with the visualization of the generated feature importance of lasso and ridge regression models.

3.6 Decision trees

The decision tree (DT) is a tree-like supervised ML model, which can be utilized for both regression and classification tasks. In a DT classification problem, the DT is going to predict a discrete value and have a discrete category in the leaf. Whereas in a DT regression problem, a continuous value will be the prediction [34] [42]. There are various DT algorithms such as ID3 (Iterative Dichotomiser 3), C4.5 (the successor to ID3), and CART (Classification and Regression Trees) [35].

Both ID3 and C4.5 are developed by Ross Quinlan. The ID3 is used for classification problems with categorical features. The basic structure of the ID3 algorithm is iterative. It generates a top-down multipath tree, which recursively divides the features into multiple groups in a greedy approach [36]. It is suitable for datasets that contain large amounts of objects, but the tree might be overfitted when training with a small sample-sized dataset [37]. The C4.5 is an extension of the ID3, and it improves upon ID3 by handling continuous data, allowing unknown values, considering attribute weights, and pruning the tree after creation [38].

The CART refers to classification and regression trees. It is similar to C4.5, but it is unlike in that CART supports regression problems [35]. In DT, a root node represents the

entire features being analysed, an internal node shows one of the splitting rules based on an attribute, and a leaf node (terminal node) represents a specific division area determined by the splitting rules along the way [38] [39]. The CART creates a binary DT by dividing the features into certain subsets at each node and making a prediction by moving from the root node to the leaf node [38].

In this thesis study, the DT regressor from the scikit-learn library is used to construct a tree-like structure and make predictions for numerical output. In addition, the implementation of DT regressor in scikit-learn uses an optimized version of the CART algorithm [35].

For the CART regression tree, it finds the best-split condition that minimizes the impurity of the child nodes the most [40]. The commonly used impurity functions include the sum of squared deviations about the mean, the sum of squared prediction errors, and the sum of absolute prediction errors [40, 41].

```
BEGIN:  Assign all training data to the root node
        Define the root node as a terminal node

SPLIT:
New_splits=0
FOR every terminal node in the tree:
    If the terminal node sample size is too small or all instances in the
    node belong to the same target class goto GETNEXT
    Find the attribute that best separates the node into two child nodes
    using an allowable splitting rule
    New_splits+1
GETNEXT:
NEXT
```

Figure 11. Sketch of simplified tree-growing algorithm [41].

Figure 11 demonstrates a simplified algorithm sketch for DT growing without being over-complicated with the core details of the CART regression tree. The CART uses the least squares and mean absolute deviation criterion [45]. The least squares regression tree (LSRT) is one of the most used regression tree models, which chose the split by minimizing the least squares error. It begins from the root node with a list of predictors (features). It recursively tries different thresholds for one predictor and calculates the average of the squares of the residuals to pick the threshold that gives the smallest result. The minimized split threshold value with the least square option can be explained as [44]:

$$R(t) = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y}(t))^2 \quad (21)$$

where \bar{y} is the mean of the dependent variable, and n is the number of observation samples in the t th subset. The \bar{y} value at any node can be replaced by regression model $f(\theta, x_i)$ with samples $\langle x_i, y_i \rangle$, where x_i represents for a set of independent variables, y_i represents for a set of dependent variables, and θ represents for a set of parameters [44] [45]:

$$R(t) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_i))^2 \quad (22)$$

The best threshold of this specific predictor becomes one candidate option for the root node of the tree. Then the same process is repeatedly applied to the rest of the predictors and the one with the lowest error value will be picked as the root node.

After the root node is formed, the process recursively splits the node into a maximum of two child nodes based on the best predictor and split condition [46]. The splitting process reduces the impurity of the nodes, the best-split condition can be found by calculating which split options may decrease the impurity of the child nodes the most. The scikit-learn DT regressor determines the split commonly with least squares error (LSE), MSE, Poisson deviance, and MAE [35]. Then the observation samples split to the left side if a specific feature is less than the threshold and the rest to the right side. The splitting rule is applied for all the remaining nodes repeatedly until the rule no longer splits the samples into smaller groups, and then a completed regression tree is formed.

In addition, stopping the split from a node when a minimum number of observation samples is reached helps to prevent overfitting problems from the model, and the node thereby changed to a leaf node. The reason is that a deep splitting tree can fit the training data perfectly but not perform well with testing data.

The regression tree models are built with input training samples that are used to make estimations of the new sample. The model applies suitable splitting rules, and the new sample at the end reaches one of the leaf nodes. The prediction value is represented by the average value of all observation samples that are located in the specific final leaf node [44].

3.7 Random forests

The random forest (RF) is an ensemble of the DTs, that combines tree predictors $h(x; \theta_k), k = 1, \dots, K$ such that each tree is created using a random vector θ_k sampled independently of the observed input vector x [47] [48].

The accuracy of RF is improved from the created ensemble of trees by using random combinations of features at each node of a grown DT. [49]. To create these ensembles, bagging (bootstrap aggregating) is often used to train each tree with a different random dataset of examples from the training set [50]. The steps to train the RF includes [51]:

- 1) Creating a bootstrapped dataset by randomly selecting the samples of the original dataset, from where the same samples are allowed to be picked more than once.
- 2) Building the DT by using the bootstrapped dataset but only considering a random subset of features (columns) at each step.
- 3) Repeating the above two steps by bagging a new dataset and building a tree with a random subset of features at each step.

This thesis study focuses on regression problem, which randomly select a combination of features at each node to grow a tree. With regression setting, the RF prediction can be illustrated as the unweighted average over the created DTs collection [47]:

$$h(x) = \frac{1}{K} \sum_{k=1}^K h(x; \theta_k) \quad (23)$$

The new observed sample passes through all the trees created by RF with the bagging technique and aims for a numerical outcome. Typically, about 67% of the original dataset is used for bagging and put into the bootstrapped dataset. The remaining 33% of the data, which are not located in the bootstrapped dataset, are called out-of-bag (out of the bootstrap sampling) [48]. The out-of-bag samples can be put into the model and passed through all the trees that were created without them. The incorrect prediction compared to the ground truth is called out-of-bag error. In addition, the predicted outcome for the classification problem will be the most voted option as a result.

4. DATA PREPARATION

This chapter includes data collection, data preparation, exploratory data analysis, and feature engineering.

4.1 Data collection

As shown in Figure 12, the target product category collected for this study is the pipe clamp which includes one pipe-type and two pipe-type. The key cost drivers and attributes related to the target product category are identified by experts with the expert judgement approach. After identifying an appropriate range of the relevant variables, the determined data fields or columns are gathered from various tables or views of data warehouses with the consideration of schema and data structure. The original upper data sources include an Oracle-based data warehouse and a PostgreSQL-based data warehouse.

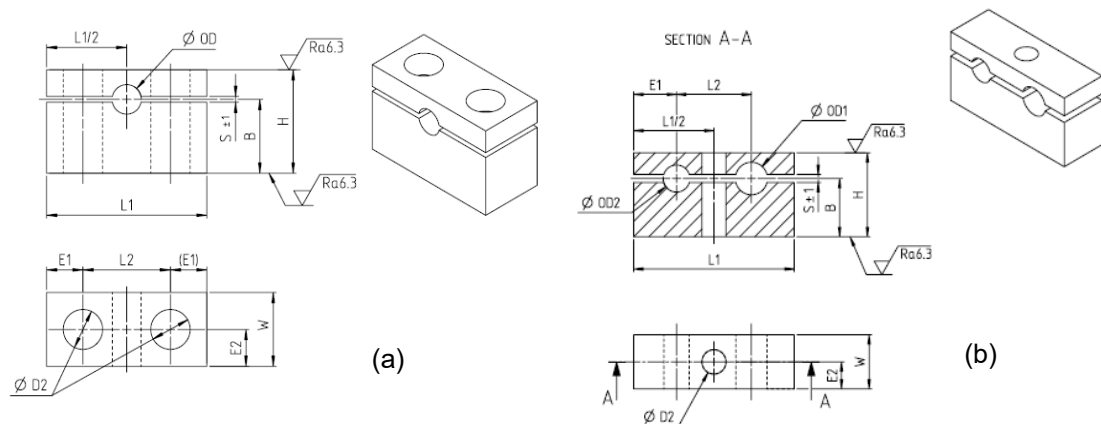


Figure 12. Pipe clamp from engineering drawing. (a) Pipe clamp with one pipe-type. (b) Pipe clamp with two pipe-type.

The data used in this study are collected from the above-mentioned data warehouses. In order to sort out the data, several steps are undertaken for this thesis analysis. First, the datasets are gathered from multiple databases of the case company, and then the data types are transformed and aligned into the correct ones. Finally, the datasets are merged into a unified data format for conducting further experiments and analyses. For this study, the Pandas DataFrame data structure is selected for processing the retrieved data. The retrieved data contains two entities: (1) the characteristics of the target product category (shown in Table 1), and (2) the class attributes of the target product category (shown in Table 2).

Table 1. Variables table based on characteristics of target product category.

Variables	Type	Description
MAT_ID	Categorical	Material ID
AGREEMENT_NO	Categorical	Agreement number in SAP
ORDERED_PCS_12_MONTHS	Numerical	Ordered in the past 12 months
AGR_UNIT_PRICE	Numerical	Agreement unit price
PO_NET_PRICE_PER_PC	Numerical	Purchase price per piece
MOVING_AVERAGE_PRICE	Numerical	Moving average price
SCALE_PRICE_EUR_PER_PC	Numerical	Scale price per piece in EUR
SCALE_QUANTITY	Numerical	Scale quantity
VEND_ID	Categorical	Id of vendor
VEND_NAME	Categorical	Name of vendor
COUNTRY_ID	Categorical	Country ID
PURCHASER	Categorical	Purchaser
PURCHASING_CATEGORY	Categorical	Purchasing category
CATEGORY_MANAGER	Categorical	Category manager
LOCAL_CURRENCY_UNIT	Categorical	Local currency unit
MAT_NAME	Categorical	Material name
MATERIAL	Categorical	Material description

DIMENSIONS	Categorical	Dimension of item
WEIGHT_NETTO	Numerical	Component net weight
SHOULD_COST	Numerical	Calculated should-cost
PLANT_ID	Categorical	Plant ID

Table 2. Variables table based on class attributes of target product category.

Variables	Type	Description
S	Numerical	Slot width
E1	Numerical	Bolt hole positioning distance from left surface
E2	Numerical	Bolt hole positioning distance across width
H	Numerical	Overall height
B	Numerical	Pipe hole centre distance from bottom surface
L1	Numerical	Overall length
L2	Numerical	Distance between two bolt holes centreline
D2	Numerical	Bolt hole diameter
OD	Numerical	Pipe hole diameter
OD1	Numerical	Pipe hole diameter (first from right side)
OD2	Numerical	Pipe hole diameter (second from right side)
W	Numerical	Width

4.2 Data cleaning

Data cleaning is a process for addressing and correcting any mistakes from the dataset to ensure the data is in high quality and suitable for analysis. The common options to handle the data cleaning include: (1) ignoring or deleting the mistake data; (2) replacing or correcting the missing value; and (3) imputing with reasonable value [53].

The data density of the collected product category dataset is visualized by using Missingno, which is a Python package designed for visualizing missing values and visually identifying the data completion [52]. The features *MAT_ID*, *MAT_NAME*, *DIMENSIONS*, and *WEIGHT_NETTO* appear to be dense as they are the basic information of an item. However, other features of characteristics and class attributes exhibit significant missingness.

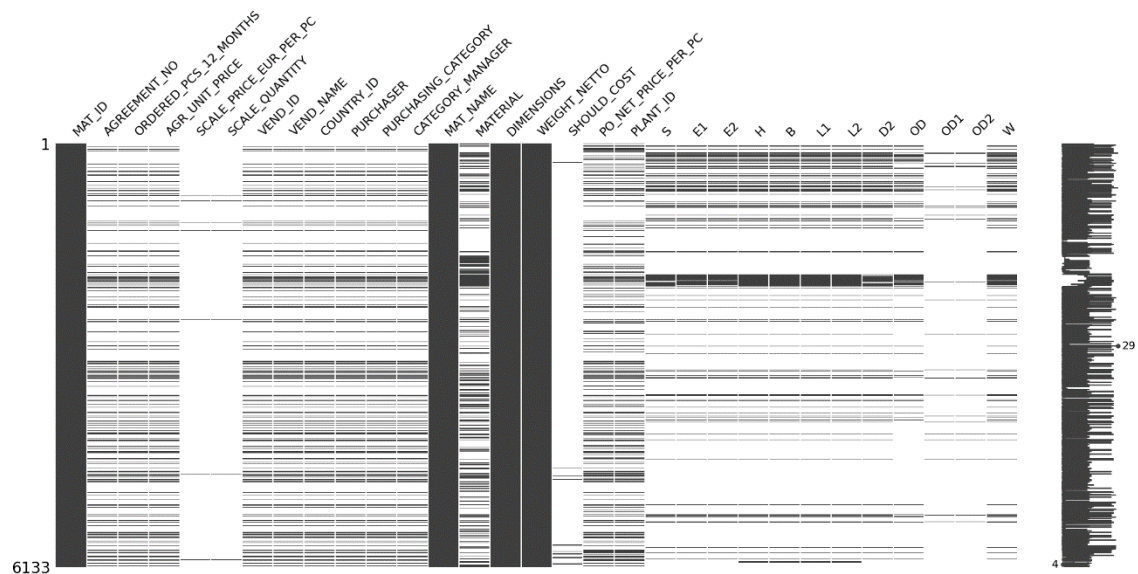


Figure 13. The data-dense display created with Missingno.

The data-dense display illustrates the extreme sparseness of the collected data in Figure 13, which contains many missing values in several selected variables. In general, there are three missingness patterns for the missing value imputation: (1) missing completely at random; (2) missing at random; and (3) missing not at random [54]. For this study, the missing values are considered as “missing at random” because the characteristics of a product can be without any information and the class attributes of a product can be also missing.

Data imputation is a common way to fill in missing value, which can potentially improve the quality of a sparse dataset with proper imputation methods. There are several native imputation methods which include [55] [54]:

- 1) Filling the missing value with a known value such as zero.
- 2) Filling in missing values with column-wise mean or median for numerical columns and most frequent value for categorical columns.
- 3) Imputing missing value by K Nearest Neighbors of the observation.

For this study, the missing values of a specific product category cannot simply be imputed by means or probabilities. This is because the technical values may not be suitable for these methods. Therefore, the rows-wise missing values are considered for removal. The further analyses are based on the data which are possible to use. However, the limited data amount of the target product category after data cleaning may potentially cause the problem of overfitting, wherein the trained model may show strong performance for smaller known data but result poorly with larger unknown data. For this reason, the cross-validation technique is used to prevent overfitting in a small dataset.

4.3 Feature engineering

Feature engineering in ML models aims to design smart features by adjusting the existing features with transformation techniques or creating new features for modelling. It helps the ML algorithm to understand the input features and it is commonly utilized after data collection and data cleaning process [56]. The feature in ML represents a certain property of the observations and can mainly be divided into numerical features and categorical features. In Figure 14, the most commonly used normalization techniques for numerical features are linear scaling, clipping, log transformation, and Z-score [57].

4.3.1 Transforming numerical feature

The linear scaling transforms numerical feature values from the actual range to a standard range, and the transformed range is typically between 0 and 1 (or -1 to +1) to

ensure that all features are located on a fixed scale. The equation of linear scaling can be explained as [57]:

$$X_{scaled} = (X - X_{min}) / (X_{max} - X_{min}) \quad (24)$$

where the X refers to the feature in the dataset. The scaling can be applied when the maximum and minimum bounds of the dataset are identified, and the dataset exists only a few outlier observations.

The feature clipping is used in normalization to handle extreme outliers by capping all extreme values above or below a specific value to a fixed value. For instance, clipping can be performed and set to exactly 35, when the collected values of a room temperature feature exceed 35 degrees Celsius. The log (logarithm) scaling is used to transform the numerical feature to its logarithm value, which is commonly applied when the distribution of a specific feature conforms to power law [57]:

$$X_{log} = \log_b(X) \quad (25)$$

where the b represents the logarithm base, and X_{log} is the transformed value.

The Z-score represents the number of standard deviations of a particular data point from the population mean. It can be used when feature distribution contains only a few extreme outliers, and the formula can be explained as [57]:

$$X_{scaled} = (X - \mu) / \sigma \quad (26)$$

where μ represents the mean and σ represents the standard deviation.

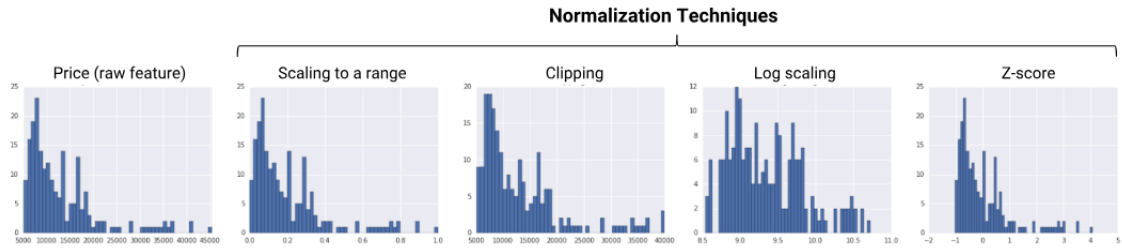


Figure 14. Summary of normalization techniques [57].

4.3.2 Transforming categorical feature

The categorical features in the collected dataset are variables that represent discrete entities [58]. These features do not behave in an ordered numeric relationship, and each unique value of a categorical feature is called a category [59]. The common encoding methods contain one-hot encoding, target encoding, and label encoding.

The one-hot encoding represents each category in a categorical feature as a vector. The encoding process can be explained as follows. Assuming x is a discrete categorical feature with n unique values x_1, x_2, \dots, x_n . The encoded vector for value x_i contains zero for every category except the i th category, which contains one as a value [61]. For instance, an arbitrary categorical feature contains values ‘a’, ‘b’, and ‘c’. The one-hot encoding can transform these values into $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$.

The target encoding transforms the observation value in a certain categorical feature based on the mean value of the target variable [60]. While the label encoding method tries to replace every category value of a categorical feature with any distinct value [61]. For instance, a categorical feature with values {“red”, “green”, “blue”}. The label encoding can respectively transform the data with arbitrary values such as $\{1, 2, 3\}$.

This thesis study aims for regression tasks, and the selected regression models generally require numerical features as input. Thus, encoding the categorical features is needed to transform the features into a numerical format.

4.4 Data exploration

This sub-chapter discusses data exploration, which is one of the key aspects of discovery-oriented applications. It facilitates the extraction of knowledge from datasets efficiently [62]. The data exploration processes included different methods, the ones applied in this thesis study include: (1) descriptive statistics, with the measurement of central tendency (for instance, count, mean, and standard deviation); (2) visualization, for the analyses of numerical feature distributions by using suitable tools (for instance, existing libraries Seaborn, Missingno and Matplotlib); and (3) statistical analysis, with Pearson correlation for revealing pairwise linear relationships. These approaches advance the understanding of the relationship between features and the distribution of the collected dataset. These are also supportive of identifying potential issues that need to be noticed.

4.4.1 Data distribution

In Figure 15, the summarized data distribution for each numerical feature is illustrated by using a histogram plot. It revealed the main located samples, the possible extreme outliers, and the best suitable normalization method to apply. The most frequently observed values for features *ORDERED_PIECES_12_MONTH*, *SCALE_PRICE_EUR_PC*, *OD1*, *OD2*, *SHOULD_COST*, and *SCALE_QUANTITY* are concentrated close to value zero, which means that the particular feature either had many missing values or no data was recorded. The trend of feature *WEIGHT_NETTO* satisfies the power law and is suitable with the log scaling method to transform the values into a narrower range. The Z-score can be applied to most of the class attributes of the target product category. This is because these values are recorded from engineering drawings that are assumed as reliable references. Hence, they are not considered to contain extreme outliers.

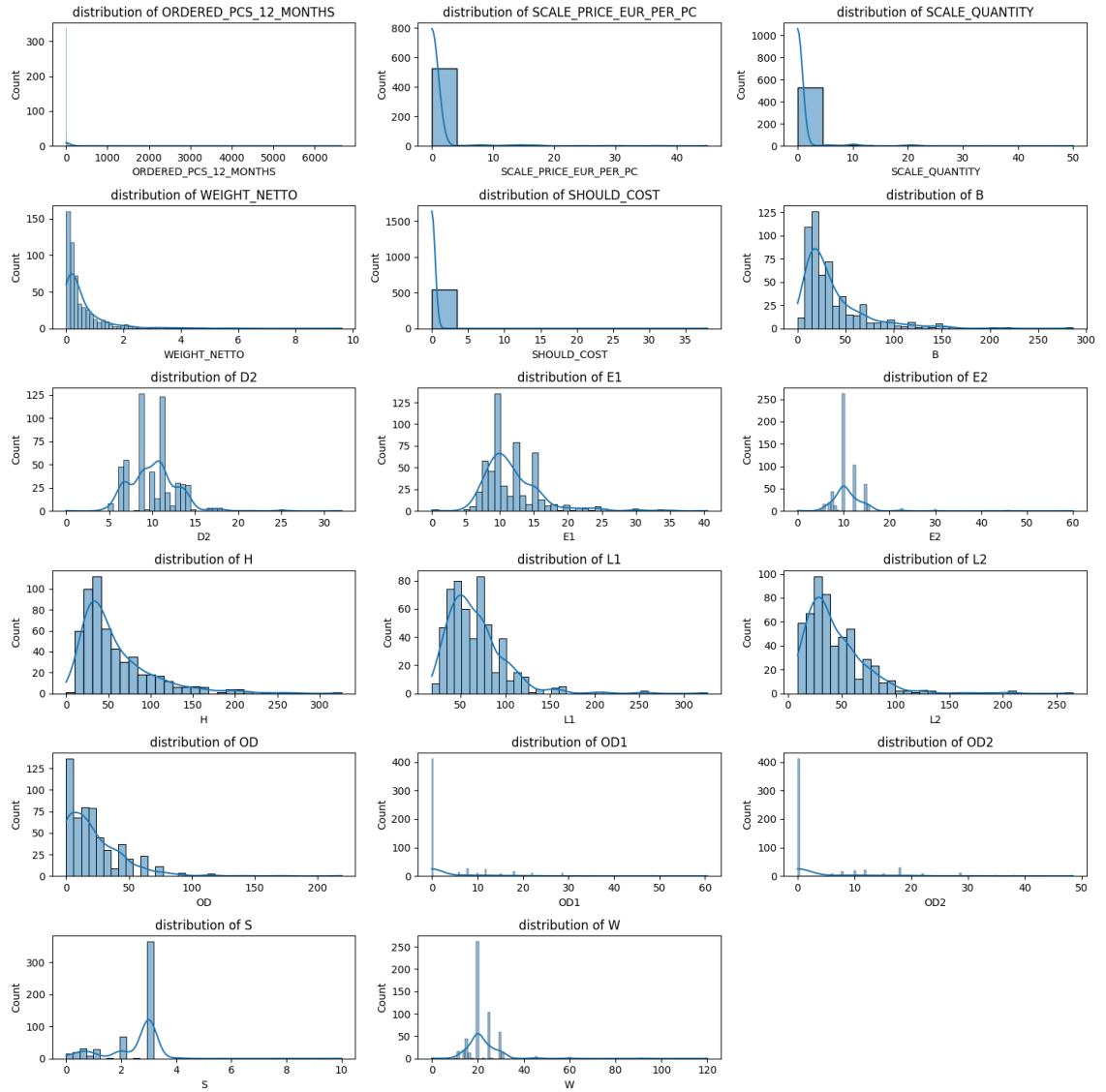


Figure 15. The data distribution of numerical features.

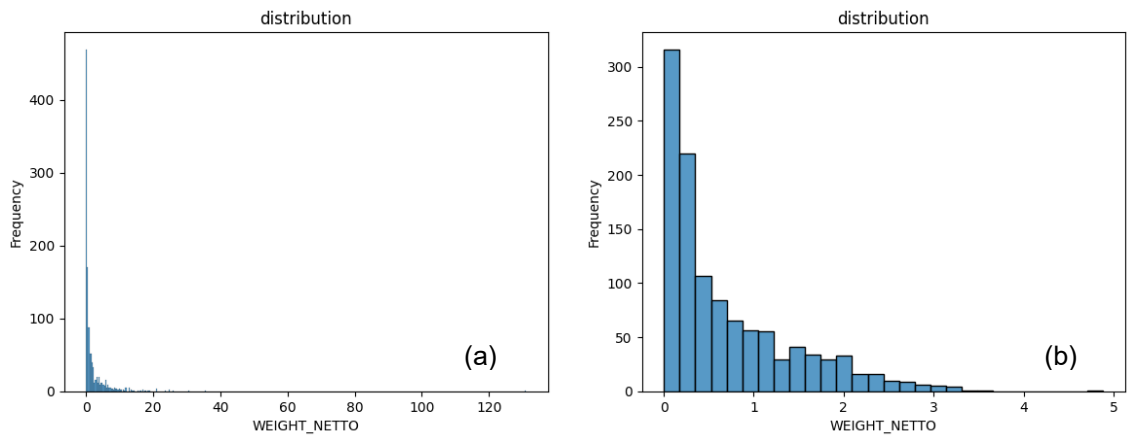


Figure 16. The data distribution plot of `WEIGHT_NETTO` feature. (a) Before log normalization. (b) After log normalization.

Figure 16 shows that the log normalization was applied to the *WEIGHT_NETTO* feature, and the range of sample points is transferred into a narrower range. The linear relationship between the *WEIGHT_NETTO* feature and the target variable may be changed after transformation. This is because the nonlinear logarithm transformation can affect the measured Pearson's correlation between features.

4.4.2 Pairwise comparison

The group of scatter plots shown in Figure 17 demonstrates the pairwise relationship between all the numerical features (predictors) and the target variable. For most of the numerical features, the majority of sample points locate in the left bottom corner of the plots. This reveals that the collected dataset contains extreme outliers, which deviate significantly from the overall data pattern. Thus, it is crucial to remove the outliers to find a stronger linear relationship and reduce the outlier impact on the linear model. In addition, the target variable represents the possible product price to be predicted.

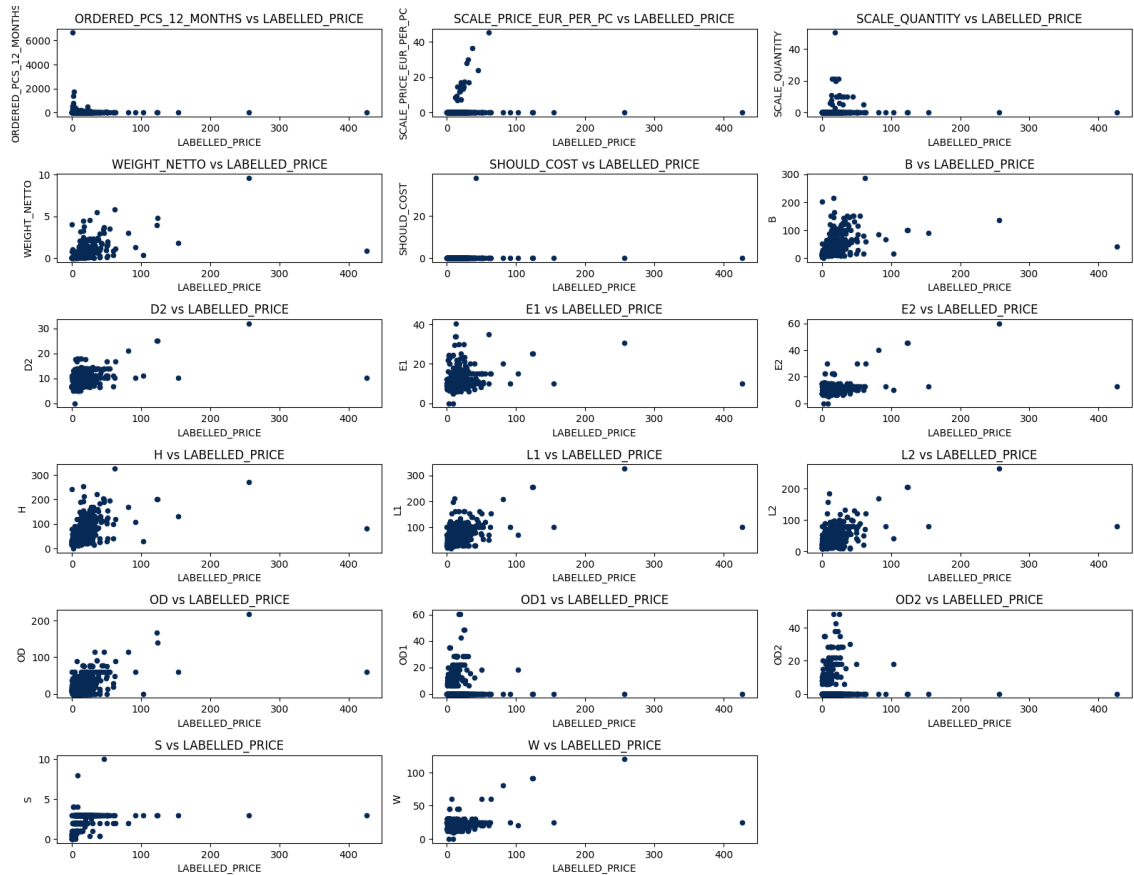


Figure 17. Scatter plot between numerical features and the target variable.

4.4.3 Outlier filtering

The outlier filtering is applied to this study to improve the reliability of data, strengthen the linear relationship between features, and enhance the accuracy of the predictive mode. As the class attributes are recorded based on the engineering drawings, the values cannot be regarded as outliers even if there may be a gap between maximum and minimum values. Thus, the outliers filtering is first considered from the target variable. The ideal target variable is the should-cost of the product, but the calculated and recorded amount of the should-cost is very small. For this reason, the possible price of the product is formed with priority of:

- (1) agreement prices prioritized by the most ordered one.
- (2) valid price from purchase order prioritized by specific plants.

(3) moving average price prioritized by specific plants.

By this approach, in the case where data for one of the options is missing, the target variable can be formed by utilizing another available value from the next option.

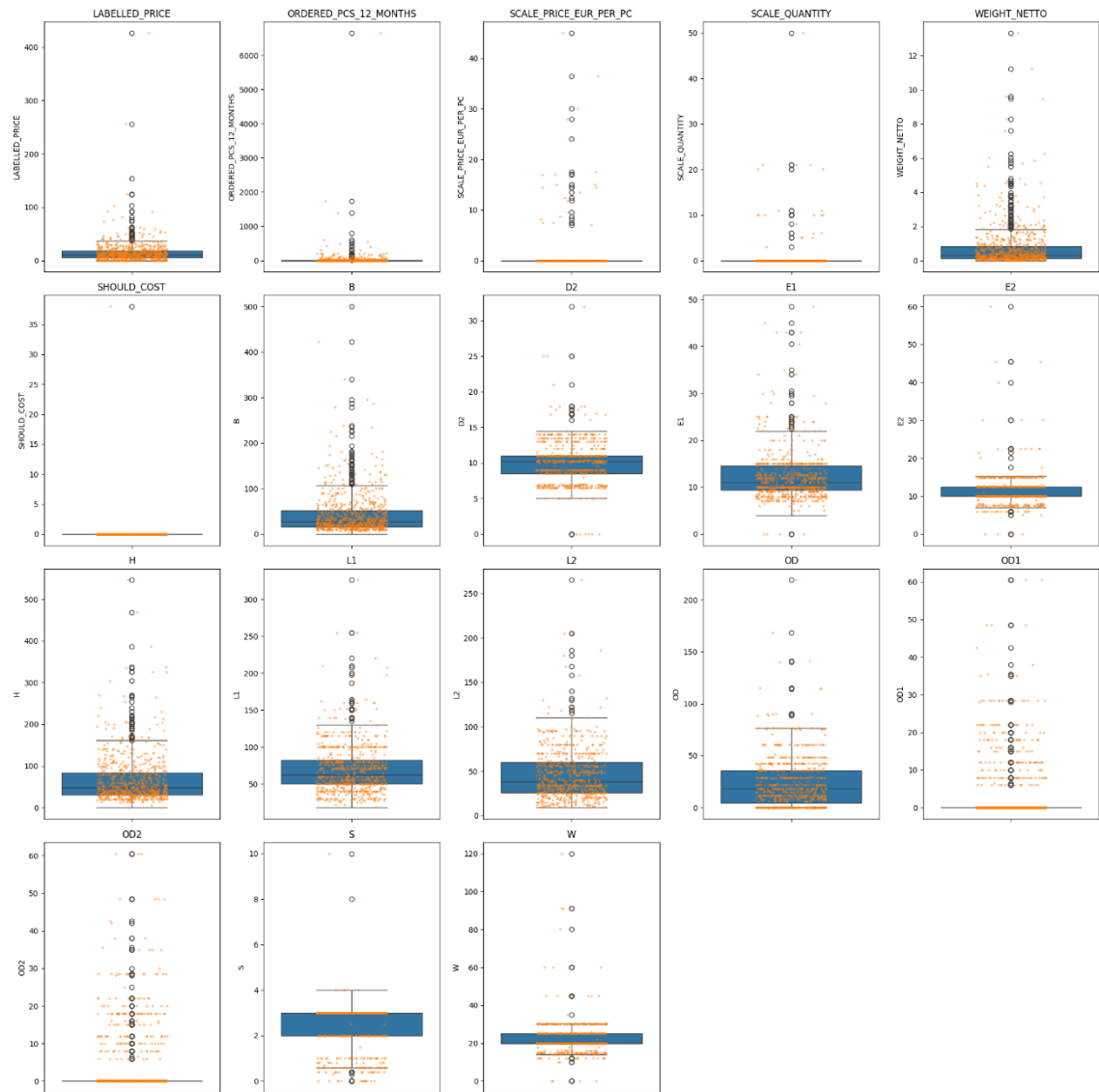


Figure 18. A set of numerical features with boxplot (blue) and strip plot (yellow).

The boxplot visualizes the spread of features, and it is commonly used for outlier detection. The median of a particular feature in a boxplot is presented by a horizontal solid line, it is bounded with the first quartile and the third quartile to form a box with 50% of the observation points included. The circle points located above the upper drawn line or below the lower drawn line are identified as outliers [63]. As it is shown in Figure 18, it

contains a group of boxplots based on all the numerical features. It shows that most of the samples in the dataset are one pipe-type pipe clamp because the feature *OD* appears a higher density of data points than *OD1* and *OD2*. As shown in Figure 19, it determines the necessity for outliers filtering in the target variable *LABELLED_PRICE*.

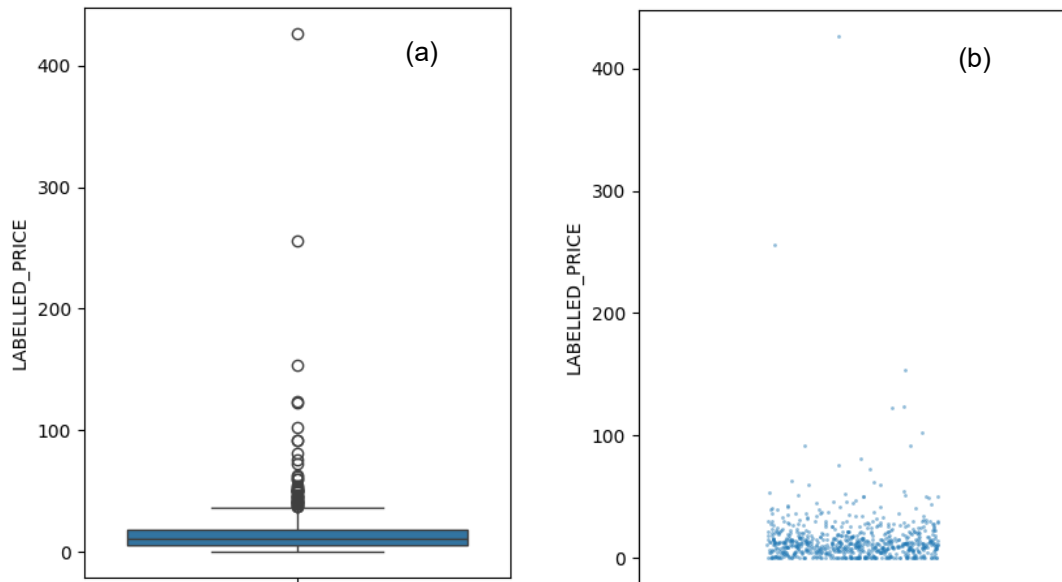


Figure 19. Identify outliers from the target variable. (a) A boxplot for the target variable. (b) A strip plot for the target variable was applied with 0.2 jitters to reduce overplotting.

Table 3 illustrates that after removing the outliers for feature *LABELLED_PRICE* with values greater than 50, some numerical features show the improvement of the pairwise correlation with the target variable (such as features *WEIGHT_NETTO*, *H*, and *B*), whereas some other features also show the decrease in pairwise correlation (such as features *E2* and *W*).

Table 3. The pairwise correlation between features and the target variable.

Features	Before outlier filtering	After outlier filtering
ORDERED_PCS_12_MONTHS	- 0.055911	- 0.135263
SCALE_PRICE_EUR_PER_PC	0.093204	0.140376
SCALE_QUANTITY	0.037669	0.117108
WEIGHT_NETTO	0.485417	0.917620

SHOULD_COST	0.043230	0.190473
B	0.336036	0.701981
D2	0.361437	0.434531
E1	0.200362	0.292089
E2	0.385305	- 0.073045
H	0.413095	0.779435
L1	0.453118	0.637182
L2	0.466977	0.623788
OD	0.476758	0.483582
OD1	- 0.029516	0.039226
OD2	- 0.029112	0.066591
S	0.224318	0.505241
W	0.387028	- 0.065984

As shown in Figure 20, a pairwise correlation matrix after outlier filtering was plotted with the Seaborn heatmap function. The Pearson correlation coefficients between pairs of variables ranging from -1 pass through 0 to 1, and the correlation coefficients illustrate the strength of a linear pairwise relationship. A negative correlation coefficient specifies an inverse relationship between two variables, meaning that the increase of one variable can result in the decrease of the other variable, and vice versa. Moreover, small datasets are generally required to have larger correlation coefficients (close to -1 or 1) in order to have significant linear relationships, while weak correlation coefficients in larger datasets can be also meaningful even though the linear relationships are limited [64]. The description of the correlation coefficient does not have a clear cut-off point, but it is commonly categorized as “weak”, “moderate”, and “strong” [65]. For instance, a correlation coefficient of 0.58 between *WEIGHT_NETTO* and *OD* shows a moderate linear association, and it suggests that when *WEIGHT_NETTO* is increasing, *OD* tends to increase slightly. Conversely, when *WEIGHT_NETTO* is decreasing, *OD* also tends to decrease.

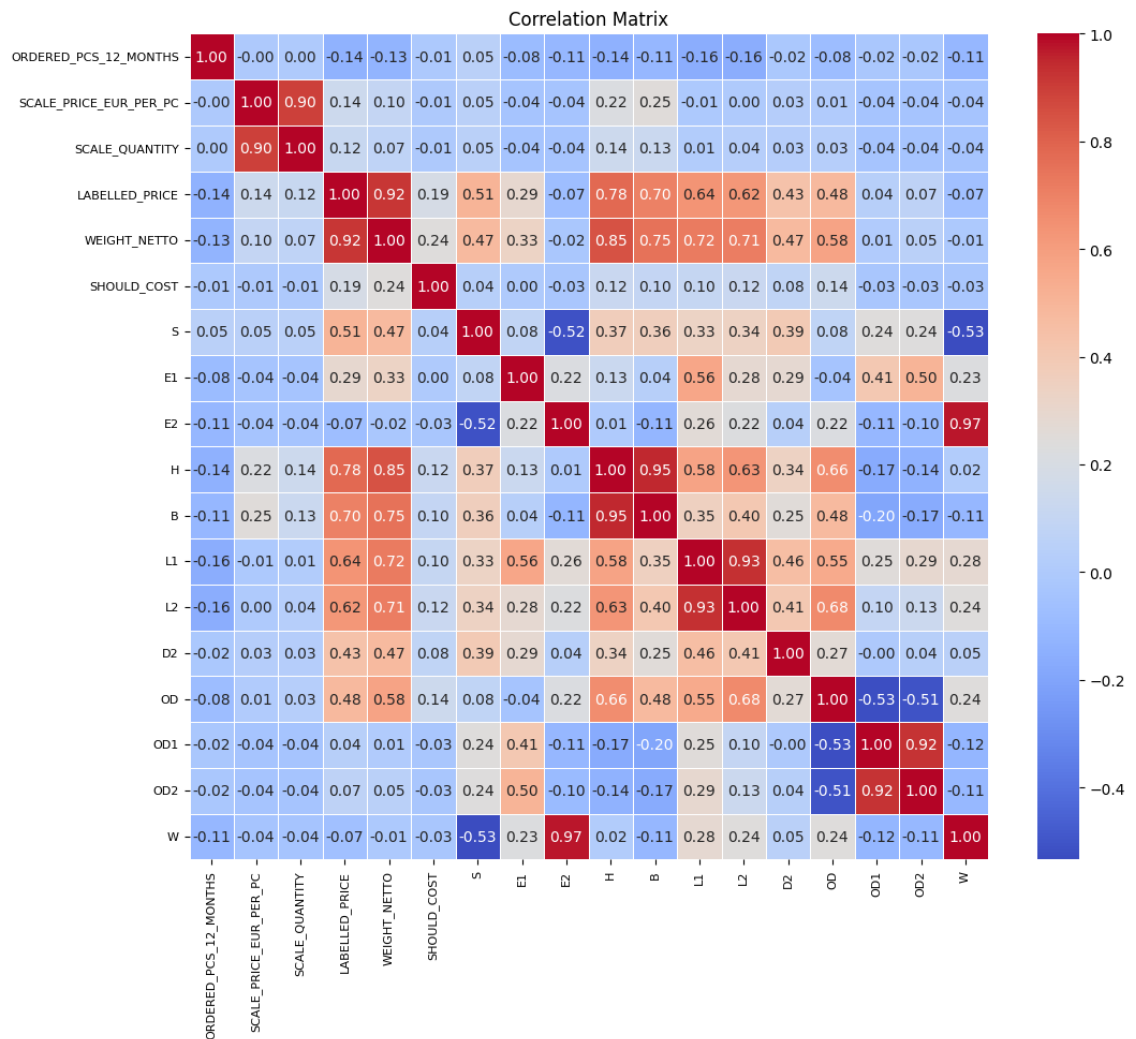


Figure 20. Correlation matrix between features with 2 decimal places after outlier filtering

4.5 Feature Selection

Feature selection concentrates on selecting the subsets of variables, that efficiently describe the usefulness of input variables and eliminating irrelevant ones. It improves the prediction performance of the ML model, reduces computation time, decreases the dimensions of data, and offers better interpretability [66]. The common feature selection methods include three categories: wrapper, filter, and embedded method. The wrapper

method uses a black box predictor to evaluate the best features from the variable subsets. The filter method utilizes the variable ranking as a selection mechanism to exclude the least promising variables. The embedded method includes variable selection in the model training process to improve efficiency [67].

In this study, the target variable is generated based on the prioritized order of *PO_NET_PRICE_PER_PC* (denoted as PO in Table 4), *MOVING_AVERAGE_PRICE* (denoted as MAP), and *AGR_UNIT_PRICE* (denoted as AGR).

Table 4. The correlation coefficients between PO, MAP, and AGR.

	PO	MAP	AGR
PO	1.000000	0.751863	0.531290
MAP	0.751863	1.000000	0.458686
AGR	0.531290	0.458686	1.000000

The agreement price is based on the formal contract that is made between the purchaser and the supplier, the purchase order price is based on the actual purchase price while the order is made, and the moving average price is based on the average price at different times. The computed correlation coefficients show a strong positive correlation between PO and MAP, a moderate positive correlation between PO and AGR, and a relatively weak positive correlation between MAP and AGR. As they all relate to the label price, thus a custom condition is applied to selectively keep only one at the end to reduce the collinearity concerns and reduce the amount of highly correlated features. Because the PO, the MAP, and the AGR are represented in similar aspects, they would be expected to have a strong positive correlation coefficient with each other. However, the actual observation contains weak correlation coefficients. This potentially exhibits the inconsistencies in the recorded data.

In Table 5, it shows the count of unique categories in a categorical feature. Intuitively, the *VEND_ID* and *VEND_NAME* are represented for the same aspect and have the

same count, thus features that represent the same thing have been dropped. The *MAT_NAME* feature which contains one unique category “pipe clamp”, was previously used as a pre-condition to extract the data with the same category, and it is now eligible for removal as there is no additional product category present at the current stage. The *DIMENSIONS* feature contains an excessive number of unique categories within a relatively small dataset, which might lead to sparse data distribution after encoding techniques. For this reason, it is removed for analysis till a larger dataset is gathered or a rule-based categorization is recognized. The rest of the categorical features are suitable for both one-hot encoding and target encoding based on the actual performance.

By experiencing feature selection, the irrelevant features and redundant features are removed and retained with the most relevant features. In addition, most of the descriptive features offer valuable insights for understanding the data but may have limited benefit on model training. Thus, these features are excluded from a regression task.

Table 5. Count of unique categories in a categorical feature.

Categorical feature name	Count of unique categories
VEND_ID	24
VEND_NAME	24
COUNTRY_ID	6
PURCHASER	11
PURCHASING_CATEGORY	7
CATEGORY_MANAGER	6
MAT_NAME	1
MATERIAL	11
DIMENSIONS	385
PLANT_ID	10

5. MODEL TRAINING AND EVALUATION

This chapter contains explanations of the experimental procedures for training selected regression models and evaluating the model prediction performance with multiple evaluation metrics.

5.1 Data splitting

Most of the ML models take a fixed and static train-test split setup for the dataset, which uses random subsets of the given data as training and validation sets, while the rest is utilized as the testing set [68] [69]. This approach ensures that the models are trained on one portion of the data, validated on another portion, and tested on a separate portion to evaluate the model performance with unseen data.

In this study, the randomized train-test split is implemented with the scikit-learn library to partition the dataset into a separate randomized training set and testing set for model training and evaluation. However, due to the limited size of the dataset, the K-fold cross-validation is used to regenerate the train-test splits to improve the performance and reliability.

The K-fold cross-validation divides a dataset into K equal-sized subsets. For each iteration of the procedure, the model is trained with $(K-1)$ folds and validated with the fold which excluded in training [70]. The choice of K is typical with a range of 5 to 10, and a value of $K < 5$ might cause an issue. Thereby, the value of K is taken as 5 to prevent unreliable evaluation.

5.2 Evaluation metrics

The evaluation metrics are used to assess the overall performance and correctness of the selected ML models. The scores of evaluation metrics indicate how well the trained models perform on a given input. The metrics serve as a criterion for hyperparameter tuning after the model training stage. The commonly used evaluation metrics include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), the coefficient of determination (R^2), and cross-validation (CV) based on certain criteria.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (27)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (28)$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (29)$$

The dataset with n observations y_1, y_2, \dots, y_n collectively denoted as y_i , and \hat{y}_i refers to the predicted values of the i th observation. The MSE metric explains the squared prediction error on average, the RMSE is the square root of MSE which interprets the evaluation on a better scale, and the MAE explains the absolute prediction error on average.

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \quad (30)$$

$$SS_{residual} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (31)$$

$$SS_{total} = \sum_{i=1}^n (y_i - \bar{y})^2 \quad (32)$$

The \bar{y} refers to the mean of the observations, the $SS_{residual}$ is the sum of squares of residuals, and the SS_{total} is the total sum of squares that tells the squared difference between observations and overall means. The R^2 is generally defined as:

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}} \quad (33)$$

A R^2 value of 1 indicates the best fit of the model, meaning that all predicted values match the observed values perfectly. A R^2 value of 0 works as a baseline model, which exhibits the predicted value always \bar{y} . The worse predictions occur when R^2 has a negative value [71].

5.3 Model training

The comprehensively designed pipeline to train a specific regression model is shown in Figure 21, where the regressor demonstrated with “LinearRegression” will recursively be experimented with all selected regression models. Furthermore, the feature engineering techniques are applied to numerical and categorical features after the train-test split to prevent data leakage. The pipeline orchestrates a series of processing steps which can be summarized as:

- (1) Passing input data into the subsequent process
- (2) Transforming categorical features with optimal encoding techniques
- (3) Transforming numerical features with optimal normalization techniques
- (4) Possible imputation methods for missing values handling
- (5) Passing through the rest of features
- (6) Applying logarithm transformation to target variable
- (7) Applying transformed features into the regression model
- (8) Applying exponential transformation to prediction

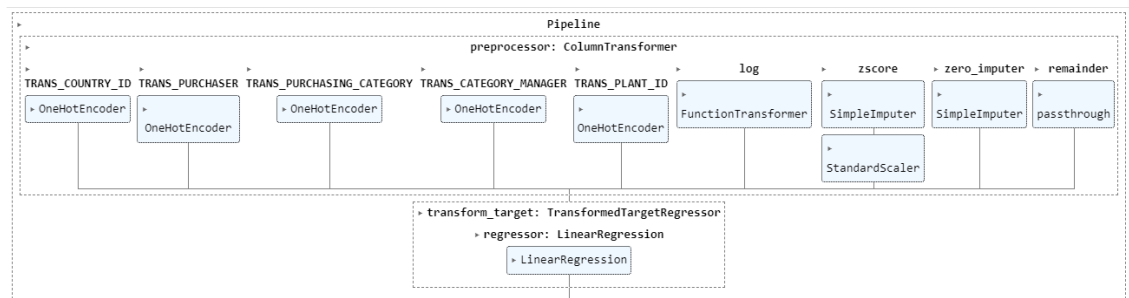


Figure 21. The designed pipeline to train with regression models

The data used in this study was collected from a particular product category (pipe clamp), which contains approximately 6000 records. However, there were a relatively smaller number of records with good-quality input variables and labelled target variables, thus, the applicable number was reduced to approximately 1000. The model was trained with 80% of the data, tested with the rest, and validated with 5-fold cross-validation. Furthermore, the logarithm and exponential transform were applied to the target variables in order to avoid negative predicted values.

The encoding techniques were first experimented with one-hot encoding. However, the one-hot encoding may significantly increase the number of new columns created by unique categories of categorical features. Generating too many new columns might be impactful for the model predictions when the dataset contains limited data. Therefore, the models were further experimentally tested with only target encoding for the categorical features. In addition, the features with the “_ENCODED” suffix represent an encoded version derived from a specific original categorical feature.

5.3.1 Exploring multivariate linear regression

The model training begins with the exploration of the MLR model, which creates relationships between multiple independent variables and a single target variable. The linear regression model used in this study was imported from the Scikit-learn library. It utilizes the least squares to find the coefficients of different predictors. The model training is first explored with default hyperparameters, followed by possible hyperparameter

tuning. The parameters for MLR to improve the model performance include: (1) The “fit_intercept” parameter (set to True by default). It determines whether to calculate the intercept of the regression line or not. If it is set to “True”, the model would estimate both the coefficients (or the slope) of the independent variables and the intercept (or the bias). Reversely, the model will not estimate the intercept term and it forces the regression line to pass through the origin point; (2) The “n_jobs” parameter (set to None by default). It indicates the number of parallel jobs to accelerate the training process, and a fixed value -1 means to use all the available processors; and (3) The “positive” parameter (is set to False by default). It works for dense arrays to force the coefficients to be positive.

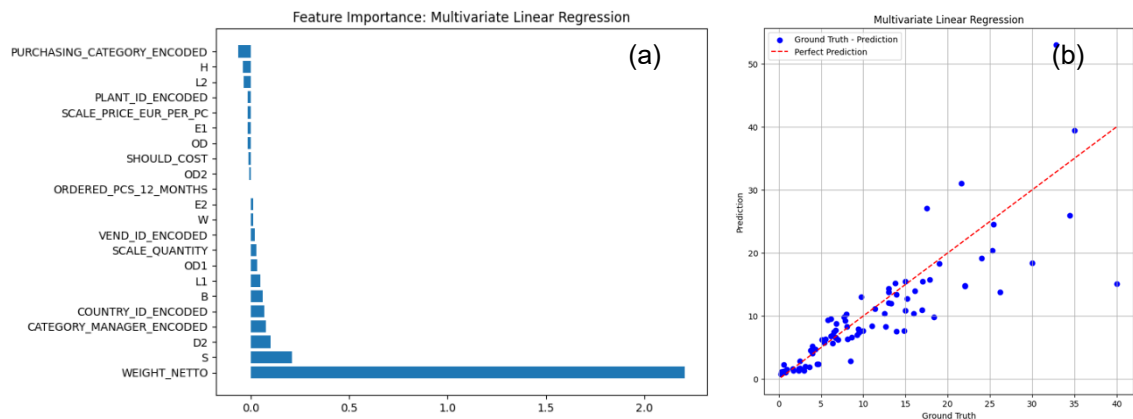


Figure 22. Exploring multivariate linear regression. (a) Feature importance. (b) Comparison between ground truth and prediction.

The coefficients in MLR represent the weights assigned to the input features. These numerical values reveal the relationship between the feature variables and the label variable. In addition, the coefficient values can be both positive and negative. Larger absolute values of the variable coefficients are supposed to cause larger impacts on the model predictions. In Figure 22 (a), the feature importance rankings were determined based on the magnitude of the feature coefficients, and the most impactful features are *WEIGHT_NETTO*, *S*, and *D2*. However, as the *WEIGHT_NETTO* exhibits the highest importance to others, it may let the model heavily rely on it to make predictions.

Both real observations and actual values can be regarded as ground truth. In Figure 22 (b), the plot between ground truths and actual predictions shows the deviation of results from the perfect line (the dotted line in the middle). It illustrates the difference be-

tween the predicted values and the actual values. In addition, the evaluation results of the MLR model (trained with default parameters) have an MSE of 26.332, an MAE of 3.306, a R^2 coefficient of 0.663, and a CV score of 0.468.

5.3.2 Exploring K-nearest neighbors regression

After applying the MLR, this thesis study continued with the exploration of the KNN model. The concept of the K-NN algorithm was explained in the earlier Chapter 3.1, which estimates the value of a target variable by analysing the average values of its K nearest sample points. The K-NN regressor used in this study was imported from the Scikit-learn library. It contains several important hyperparameters for tuning the model: (1) The “n_neighbors” hyperparameter (set to 5 by default). It determines the number of nearest points when making predictions; (2) The “metric” hyperparameter (set to “min-kowski” by default). It selects the distance function that is used to measure the similarity between sample points; and (3) The “weights” hyperparameter (set to “uniform” by default). It specifies the metrics for assigning weights to the sample points (such as uniform and distance metrics). In addition, the uniform metric considers all points that contribute equally to the prediction, while the distance metric considers those closer neighbors to have a larger impact on the prediction.

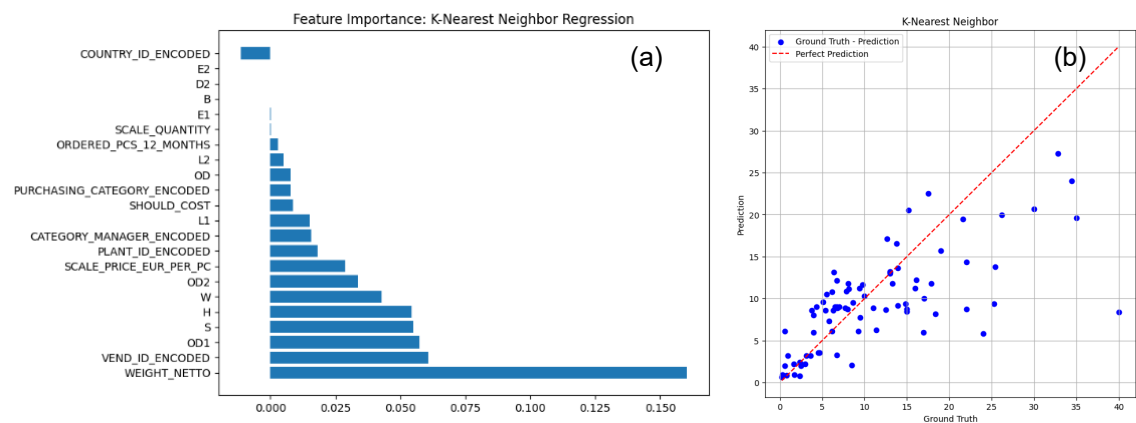


Figure 23. Exploring K-Nearest Neighbors regression. (a) Feature importance. (b) Comparison between ground truth and prediction.

The feature importance of the K-NN regressor is computed using the permutation importance technique. It assesses the importance of input features by measuring the change of results after permuting specific features in the model [72]. The most impactful features of the model are shown in Figure 23 (a), which are *WEIGHT_NETTO*, *VEND_ID_ENCODED*, and *OD1*. In Figure 23 (b), the ground truth versus prediction plot of the K-NN regressor exhibits relative dispersion between the actual and predicted values. The evaluation results of the K-NN regression model (trained with default hyperparameters) have an MSE of 41.946, an MAE of 4.273, a R^2 coefficient of 0.463, and a CV score of 0.258.

5.3.3 Exploring support vector regression

After the exploration of K-NN, it is beneficial to test with the SVR model which was mentioned in the earlier Chapter 3.3. The SVR tries to identify a hyperplane that fits the majority of the sample points within the upper and lower margins. It gives flexibility to handle how many outliers are acceptable within the margins. The main hyperparameters of SVR include: (1) The regularization parameter “C” (set to 1.0 by default). It indicates the trade-off between the margin and the error term; (2) The “kernel” parameter (set to “scale” by default). It defines which kernel function to be used in the SVR model. In this study, a linear kernel is used for the regression task, and the best-fitted hyperplane is utilized as the prediction line; (3) The “epsilon” parameter (set to 0.1 by default). It defines the tolerance of margin, where no penalty is associated with the loss function if points fall within an epsilon distance away from the actual value.

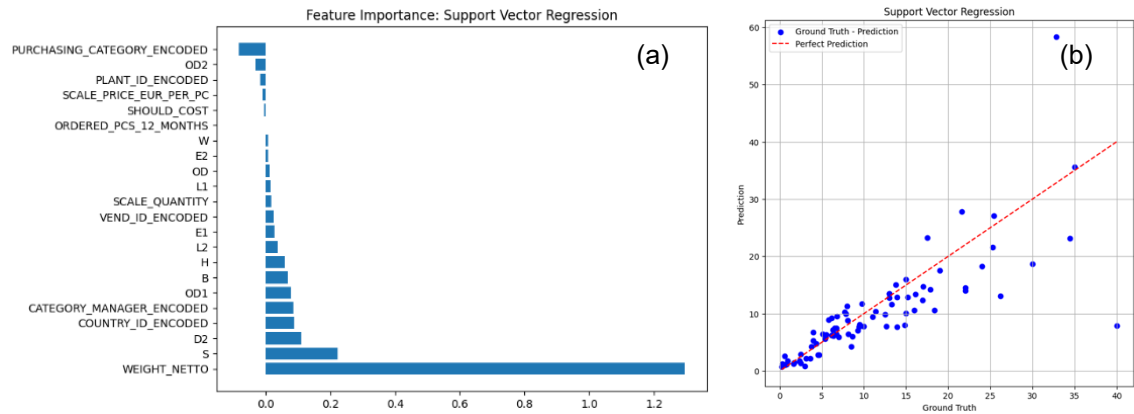


Figure 24. Exploring support vector regression. (a) Feature importance. (b) Comparison between ground truth and prediction.

The feature importance of the SVR regressor was assessed by the model coefficients associated with the input features. In Figure 24 (a), the top three features with the highest coefficients are *WEIGHT_NETTO*, *S*, and *D2*. In Figure 24 (b), the majority of the observed sample points are accurately predicted, and they closely located to the perfect prediction line. However, there happen to be some points, at which the predicted values are deviated from the actual values. The evaluation results of the SVR model (trained with default hyperparameters) have an MSE of 33.494, an MAE of 3.158, a R^2 coefficient of 0.572, and a CV score of 0.523.

5.3.4 Exploring lasso and ridge regression

As a continuation, the thesis study conducts the explorations with lasso and ridge regression, which was discussed in Chapter 3.4 and Chapter 3.5. The lasso and ridge regression are regularization techniques that are used to prevent the problem of overfitting by adding an L1 or L2 penalty term to the OLS. The lasso penalty L1 is the absolute value of the magnitude of coefficients. The ridge penalty L2 is the squared magnitude of coefficients. The lasso and ridge regressors used in this study are imported from the Scikit-learn library. In addition, they apply an “alpha” to denote the hyperparameter λ (which controls the strength of regularization). By default, the λ is set to 1.0 and it is equivalent to OLS when λ is 0.

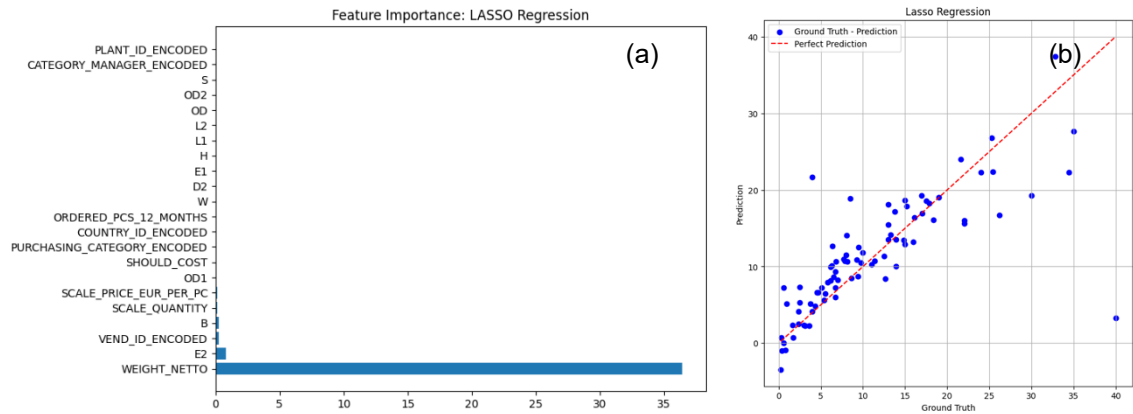


Figure 25. Exploring lasso regression. (a) Feature importance. (b) Comparison between ground truth and prediction.

Figure 25 illustrates how lasso regression shrinks the coefficients for some of the features to exactly zero to effectively exclude the less important features from the model. The features with the highest feature importance for the lasso model are `WEIGHT_NETTO`, `E2`, and `VEND_ID_ENCODED`. The evaluation results of the lasso regressor have an MSE of 33.154, an MAE of 3.252, a R^2 coefficient of 0.576, and a CV score of 0.627.

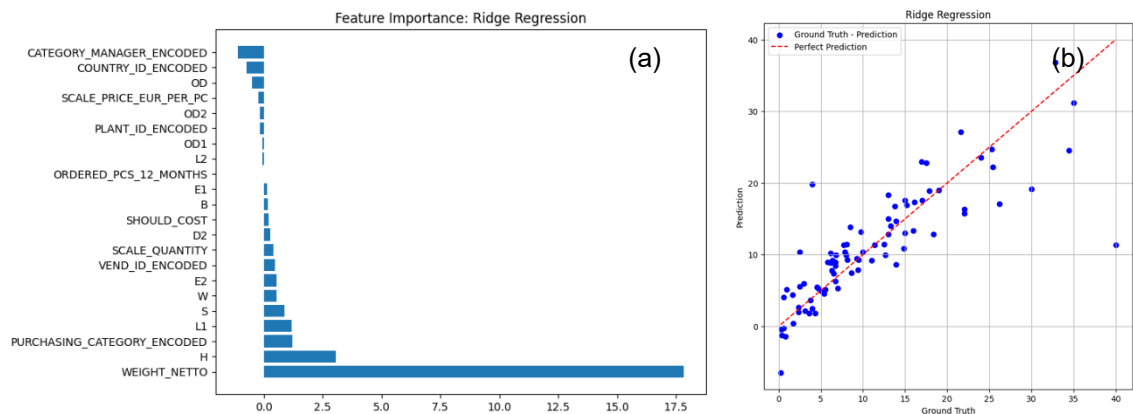


Figure 26. Exploring ridge regression. (a) Feature importance. (b) Comparison between ground truth and prediction.

Figure 26 exhibits that ridge regression retains all the input features in the model, even if the coefficients of some features shrink towards zero. The features with the highest feature importance for the ridge model are `WEIGHT_NETTO`, `H`, and `PURCHASING_CATEGORY_ENCODED`. The evaluation results of the ridge regression model

(trained with default parameters) have an MSE of 24.795, an MAE of 3.088, a R^2 coefficient of 0.683, and a CV score of 0.746.

5.3.5 Exploring decision tree and random forest regression

At the final stage, the thesis utilizes DT and RF for the exploration. The DT regression was discussed in Chapter 3.6, which recursively splits nodes into smaller partitions to minimize the impurity of the child nodes until leaf nodes are reached. The DT regressor is imported from the Scikit-learn library, which includes several important hyperparameters: (1) The “criterion” hyperparameter (set to MSE by default). It defines which function to use for measuring the quality of a split; (2) The “max_depth” hyperparameter (set to None by default). It controls the maximum depth of the tree. A deeper created tree can improve the linear relationship in the model but may also cause the problem of overfitting; (3) The “min_samples_split” hyperparameter (set to 2 by default). It sets the minimum required samples to split an internal node, which helps to prevent the node of the tree from splitting with too few samples; and (4) The “min_samples_leaf” hyperparameter (set to 1 by default). It determines the minimum required samples at a leaf node and ensures that each leaf node contains at least a minimum number of samples.

The RF regression, mentioned in earlier Chapter 3.7, is an ensemble of trees that makes predictions based on the unweighted average of the trees that are created with a random subset of features. The RF regressor includes some important hyperparameters from the DT regressor and has some additional hyperparameters: (1) The “n_estimator” hyperparameter (set to 100 by default). It determines the number of DTs that are used in the ensemble; (2) The “bootstrap” hyperparameter (set to True by default). It indicates whether bootstrap samples are used when creating each DT; and (3) The “max_samples” hyperparameter (set to None by default). It defines the amount of a randomly sampled subset selected from the dataset to train each DT estimator.

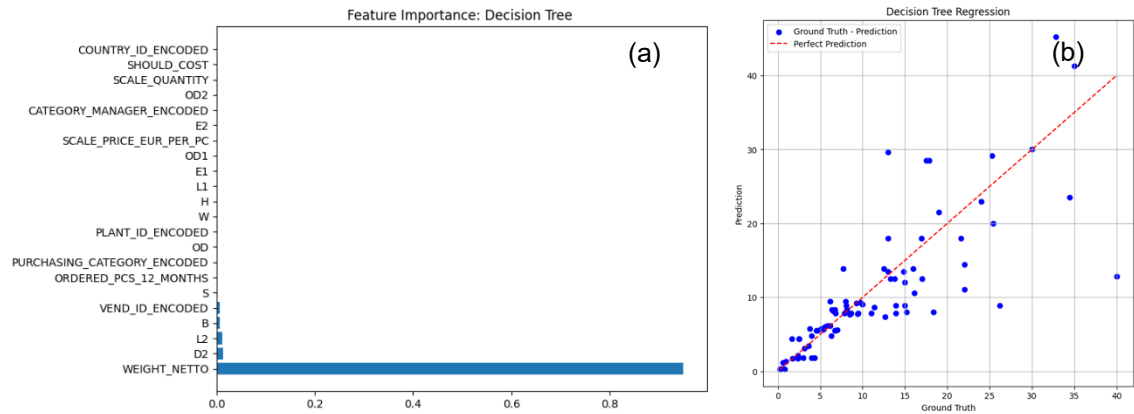


Figure 27. Exploring decision tree regression. (a) Feature importance. (b) Comparison between ground truth and prediction.

There is no coefficient associated with the features of the DT regressor. Instead, the feature importance relies on how influential a feature is in reducing the impurity of node splitting. In Figure 27 (a), the most important feature of the DT regressor is *WEIGHT_NETTO*, followed by D2 and L2 which have minor influences. Figure 27 (b) shows that the predicted values are deviated from the actual values and the trained model may not adequately capture the relationships of data. The evaluation results of the DT regression model (trained with default hyperparameters) have an MSE of 31.517, an MAE of 3.272, a R^2 coefficient of 0.597, and a CV score of 0.572.

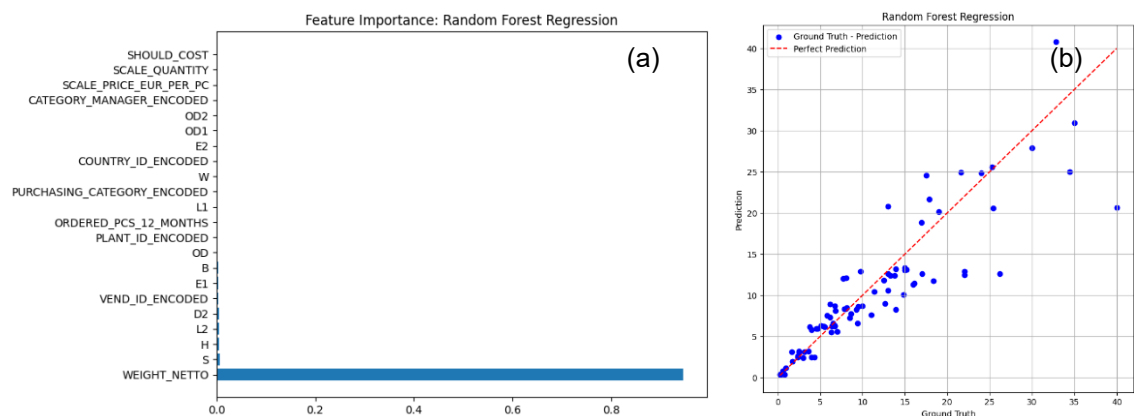


Figure 28. Exploring random forest regression. (a) Feature importance. (b) Comparison between ground truth and prediction.

As shown in Figure 28 (a), the dominant feature *WEIGHT_NETTO* is observed with the highest importance compared to the rest of the features. It may provide most of the information to the model and make the model severely rely on it without considering oth-

er weak features. Figure 28 (b) shows that the RF regressor achieves relatively improved results over the DT regressor by aggregating predictions from multiple DTs. The evaluation results of the RF regression model (trained with default hyperparameters) have an MSE of 16.657, an MAE of 2.514, a R^2 coefficient of 0.787, and a CV score of 0.786.

6. RESULTS AND DISCUSSION

6.1 Overview and comparison of model results

As shown in Table 6, the statistical summaries of the different regression models are generated based on multiple evaluation metrics. The evaluation results provide a baseline of prediction performance and illustrate how well a particular model behaves with the unseen data. The models with lower MSE, RMSE, and MAE scores, and higher R^2 and CV scores are generally considered to possess better accuracy and overall performance. This is because these metrics can reflect the reduction in prediction error between the actual and predicted values over multiple CV experiments.

The hyperparameter tuning applies to the models with the grid search technique that systematically tries a given list of hyperparameters for a particular model to identify the best combination of parameters that provides the best performance to the model. The comparison of model results before and after hyperparameter tuning is shown in Figure 29 and Figure 30. The models with the “_TUNED” suffix are applied with grid search optimization, and without the suffix means the models have not undergone hyperparameter tuning. This is because the exploration of the models is run on the cloud-based Jupyter Notebook instance and the grid search method for retrieving the best parameters often encounters session timeouts as it takes several hours for each computation.

Table 6. Evaluation results for used models

Model	MSE	RMSE	MAE	R2	CV
MLR	26.618	5.131	3.038	0.663	0.468
MLR_TUNED	26.612	5.159	3.140	0.660	0.524
K-NN	41.946	6.477	4.273	0.464	0.258
K-NN_TUNED	35.595	5.966	3.635	0.545	0.436
SVR	33.494	5.788	3.158	0.571	0.523

DT	29.577	5.439	3.075	0.622	0.578
RF	16.657	4.081	2.514	0.787	0.786
LASSO	33.154	5.758	3.252	0.576	0.627
LASSO_TUNED	13.199	3.633	2.467	0.831	0.820
RIDGE	24.795	4.980	3.088	0.683	0.746

The results in Table 6 show that LASSO_TUNED, RF, and RIDGE regression have higher CV scores ranging from 0.75 to 0.85, which indicates a strong performance. From another perspective, the rest of the models exhibit moderate CV scores ranging from 0.45 to 0.65. The evaluation score results across the selected models are various. The differentiations of the scores vary depending on: (1) the most important features applied in a particular model; (2) the quality of collected data judged in different aspects; and (3) whether the model is suitable for the problem.

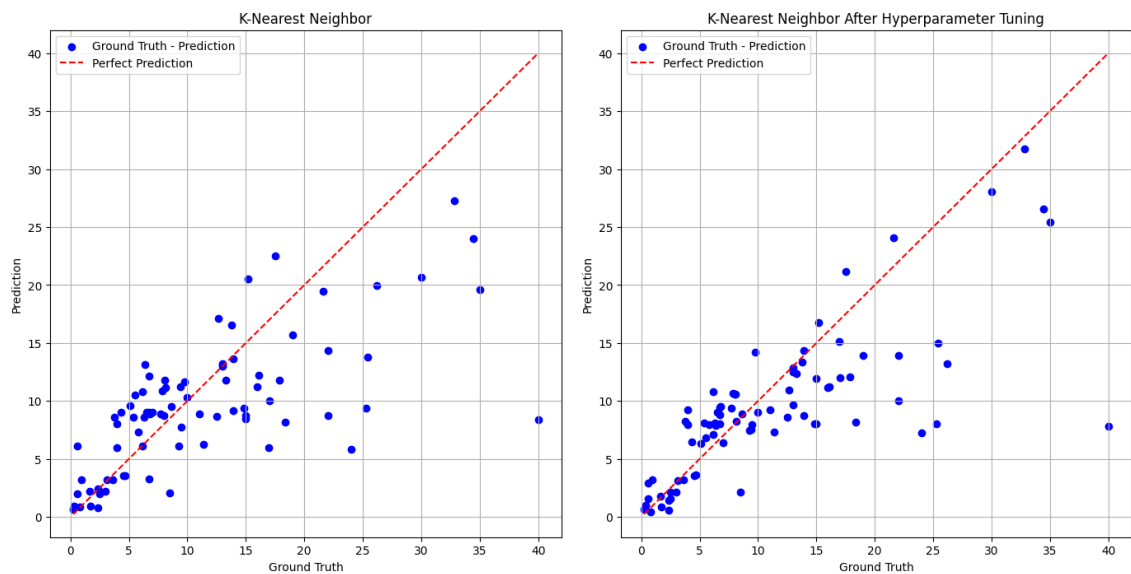


Figure 29. The comparison of K-Nearest Neighbors regression model results before and after hyperparameter tuning.

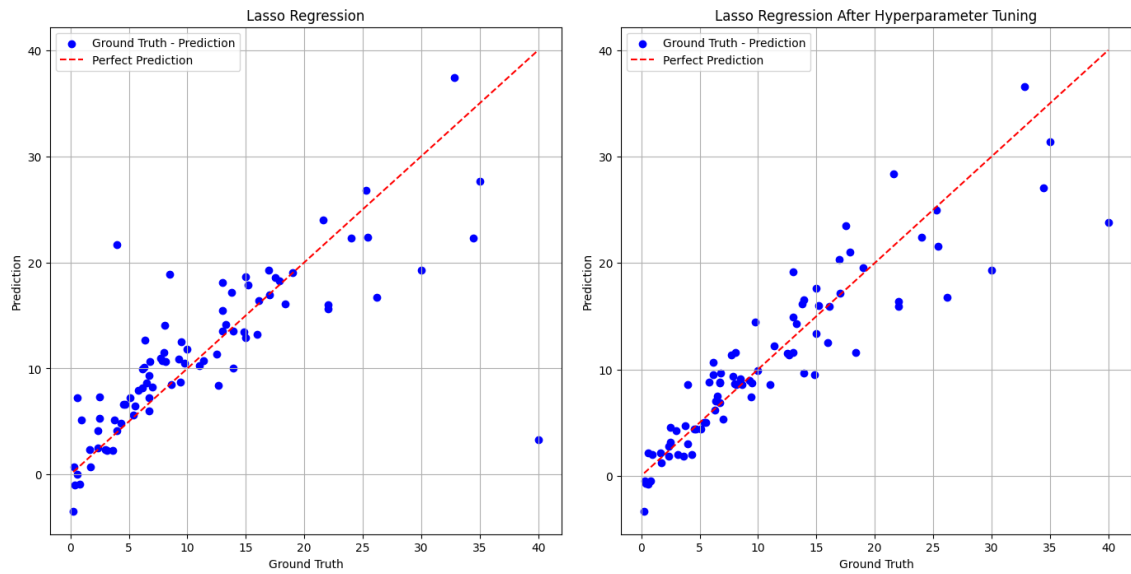


Figure 30. The comparison of lasso regression model results before and after hyperparameter tuning.

6.2 Challenges of current work

The aim of this thesis study is to make should-cost estimations for a specific product category so that the predicted values can be used as a baseline when producing new similar items. There are several challenges encountered during the thesis study.

The first challenge is the lack of literature for the product should-cost estimation. The search string results at resource databases (such as Springer, Scopus, and Web of Science) show that there is relatively small number of existing research. In addition, some of them are related to proprietary software products offered by certain companies, which cannot directly be applied in this study. Therefore, only the available resources and methodologies in this field are considered.

Another challenge is related to the size of the dataset that is used to train the regression models. The current dataset shows sparsity with both feature variables and the target variable. The lack of target variables can cause some problems because the regression tasks (which fall under supervised learning) generally require accurate target variables to learn the pattern and obtain the linear relationship of the model.

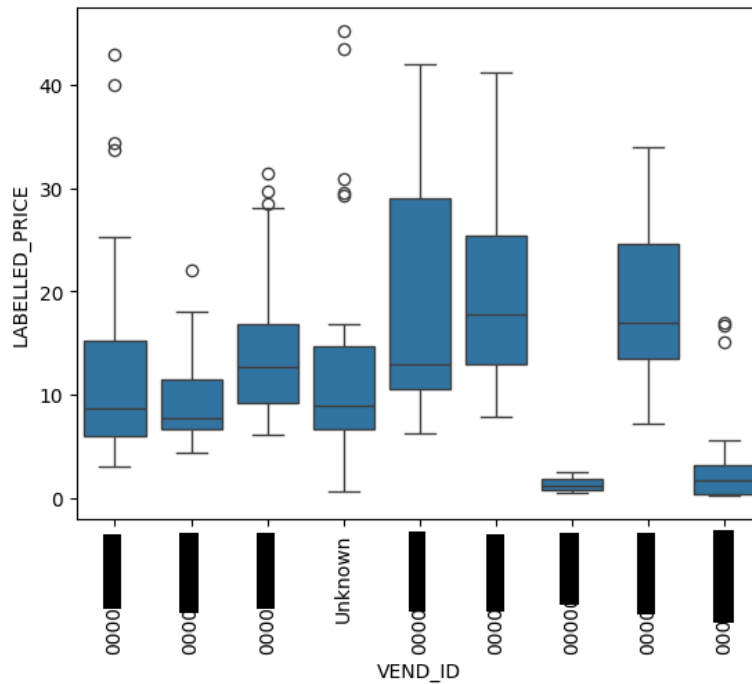


Figure 31. A boxplot of labelled prices across different vendors.

Figure 31 shows a grouped boxplot of the labelled prices across different vendors. Each box represents the pipe clamp price distribution for a specific vendor. The dataset used in the current study has a limited size in scope, thus the vendor feature is used as an input instead of a separate partition to train the models with a specific pattern.

Additionally, the existing extreme outliers in the dataset can influence the linear balance between the input features and target variables. The outliers can also to some extent disrupt the trained weight of the variables. The natural relationship between the product weight and cost for the same product category is that they are positively correlated. However, the expected relationship does not support some of the observation samples, meaning that a heavier product weight is associated with a lower price. Another example is that the label variables contain extremely small values that are unreliable based on the collected data. The above situations can affect the integrity and accuracy of the training data. Therefore, it is necessary to have more considerations of outlier filtering and actions for data cleaning during the training process.

7. CONCLUSION

The purpose of this thesis is to study the theories related to the cost estimations and generate an ML pipeline to estimate the should-cost of a product. The retrieved historical data is related to pipe clamp product category. The created pipeline runs regression models with the collected historical data to learn the relationships between feature variables and target variable. Then it is used to estimate the should-cost of an item based on the similar previously manufactured items. The most important variables involved in the training process are identified by experts in this field. As a result, integrating expert judgement along with ML methods for a particular product category can enhance the efficiency and accuracy of the cost estimation process.

The experimental results show that the lasso, ridge, and random forest regression from the selected regression models are able to capture the linear relationships in the collected dataset. It shows that different models may rely on different features by their respective feature importance rankings. This practical approach that happened during the thesis process is supposed to be suitable for making cost estimations in the early stage of product development. The approach made estimations based on similar previously manufactured items.

One of the recognized limitations of this study is the constrained size of the retrieved dataset, and the possible solution for this point is to further enhance the dataset with rule-based imputation and data cleaning. As an interpretation of the solution, obtaining more available data can advance the training models to include observation samples in the training and testing sets. In this way, the accuracy of the model training results and the trustworthiness of the evaluation scores can be enhanced. As an outcome, the results generated from the pipeline would be beneficial to strategic purchasers to determine whether the quoted prices align with or deviate from an established baseline generated by similar items.

This thesis study explored the regression-based models, which primarily require numerical features as input. It is needed to further involve experiments with the models

that handle the categorical features as input natively to offer enhanced predictive capabilities. For instance, the CatBoost model and LightGBM model provides built-in support with categorical features [73].

After the completion of the current thesis study, there are some aspects that may be useful for future work to focus on. In other words, the quality and reliability of the current study can be improved by realizing the following: (1) performing further analyses of the data based on the target product categories; (2) collecting additional datasets with a larger amount of relevant information; (3) implementing better rule-based data imputation for handling missing data; and (4) cleaning the outliers from the features that show weak importance or less accuracy.

In addition to the above-mentioned points, there may also be a necessity to investigate more potential features that give higher correlations to the target product cost. One of the options is to utilize possible technical values, for instance, stamping, punching, forming, and drilling. These values can be obtained at the late development stage of manufacturing process. Furthermore, identifying more critical factors can be useful in making more precise cost estimations, since they appear to have higher relationships to the final product.

Thus, the experimental ML pipeline established in the current study needs to be tested with other product categories to ensure reusability and validate the accuracy. Additionally, the pipeline can be further integrated into a web application to help people estimate a new product by providing the most important features of its similar products. Furthermore, the current cost estimation work can be enhanced with the existing data science platform RapidMiner to optimize the general ML workflow, the data preparation process, and the visualization of the collected data [74].

REFERENCES

- [1] Niazi, A., Dai, J.S., Balabani, S. and Seneviratne, L., 2006. Product cost estimation: Technique classification and methodology review.
- [2] Rush C, Roy R., 2001. Expert judgement in cost estimating: Modelling the reasoning process. *Concurrent Engineering*, 9(4), 271-284.
- [3] Hennebold, C., Klöpfer, K., Lettenbauer, P. and Huber, M., 2022. Machine Learning based Cost Prediction for Product Development in Mechanical Engineering. *Procedia CIRP*, 107, pp.264-269.
- [4] Ben-Arieh, D. and Qian, L., 2003. Activity-based cost management for design and development stage. *International Journal of Production Economics*, 83(2), pp.169-183.
- [5] Aram, S., Eastman, C. and Beetz, J., 2014. Qualitative and quantitative cost estimation: a methodology analysis. In *Computing in Civil and Building Engineering* (2014) (pp. 381-389).
- [6] Tayefeh Hashemi, S., Ebadati, O.M. and Kaur, H., 2020. Cost estimation and prediction in construction projects: A systematic review on machine learning techniques. *SN Applied Sciences*, 2, pp.1-27.
- [7] McKinsey, 2017. What should it cost? Available: <https://www.mckinsey.com/capabilities/operations/our-insights/what-should-it-cost> (Accessed 25.2.2024).
- [8] Machine Learning, 2024. What is Machine Learning? Available: <https://developers.google.com/machine-learning/intro-to-ml/what-is-ml> (Accessed 25.2.2024).
- [9] Machine Learning, 2023. Machine Learning Glossary. Clustering. Available: <https://developers.google.com/machine-learning/glossary#clustering> (Accessed 25.2.2024).
- [10] Machine Learning, 2023. Machine Learning Glossary. Example. Available: <https://developers.google.com/machine-learning/glossary#example> (Accessed 25.2.2024).
- [11] Bodendorf, F. and Franke, J., 2021. A machine learning approach to estimate product costs in the early product design phase: a use case from the automotive industry. *Procedia CIRP*, 100, pp.643-648.
- [12] Yoo, S. and Kang, N., 2021. Explainable artificial intelligence for manufacturing cost estimation and machining feature visualization. *Expert Systems with Applications*, 183, p.115430.

- [13] Aram, S., Eastman, C. and Beetz, J., 2014. Qualitative and quantitative cost estimation: a methodology analysis. In *Computing in Civil and Building Engineering* (2014) (pp. 381-389).
- [14] Coşkun, G.T. and Yalçiner, A.Y., 2021. Determining the best price with linear performance pricing and checking with fuzzy logic. *Computers & Industrial Engineering*, 154, p.107150.
- [15] Newman, W.R. and Krehbiel, T.C., 2007. Linear performance pricing: A collaborative tool for focused supply cost reduction. *Journal of Purchasing and Supply Management*, 13(2), pp.152-165.
- [16] Ozturk Kiyak, E., Ghasemkhani, B. and Birant, D., 2023. High-Level K-Nearest Neighbors (HLKNN): A Supervised Machine Learning Model for Classification Analysis. *Electronics*, 12(18), p.3828.
- [17] Shi, Y., Yang, K., Yang, Z., & Zhou, Y., 2021. Mobile edge artificial intelligence: Opportunities and challenges.
- [18] Hu, L.Y., Huang, M.W., Ke, S.W. and Tsai, C.F., 2016. The distance function effect on k-nearest neighbor classification for medical datasets. *Springerplus* 5 (1): 1304.
- [19] Rahim, R., Ahmar, A.S. and Hidayat, R., 2022. Cross-Validation and Validation Set Methods for Choosing K in KNN Algorithm for Healthcare Case Study. *JINAV: Journal of Information and Visualization*, 3(1), pp.57-61.
- [20] Codecademy. K-Nearest Neighbors. Available: <https://www.codecademy.com/learn/introduction-to-supervised-learning-skill-path/modules/k-nearest-neighbors-skill-path/cheatsheet> (Accessed 25.2.2024)
- [21] IBM. What is the k-nearest neighbors (KNN) algorithm? Available: <https://www.ibm.com/topics/knn> (Accessed 25.2.2024).
- [22] Al Asheeri, M.M. and Hammad, M., 2019, September. Machine learning models for software cost estimation. In *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)* (pp. 1-6). IEEE.
- [23] Khuri, A.I., 2013. Introduction to Linear Regression Analysis, by Douglas C. Montgomery, Elizabeth A. Peck, G. Geoffrey Vining. *International Statistical Review*, 81(2), pp.318-319.
- [24] PennState, Eberly College of Science, 2018. A Matrix Formulation of the Multiple Regression Model. Available: <https://online.stat.psu.edu/stat462/node/132/> (Accessed 25.2.2024).
- [25] Wikipedia, 2024. Support vector machine. Available: https://en.wikipedia.org/wiki/Support_vector_machine (Accessed 25.2.2024).

- [26] Rao, A.P., Jenkins, P.R., Auxier, J.D., Shattan, M.B. and Patnaik, A.K., 2022. Development of advanced machine learning models for analysis of plutonium surrogate optical emission spectra. *Applied Optics*, 61(7), pp.D30-D38.
- [27] Datta, R. and Singh, S., 2021. Artificial intelligence in critical care: Its about time!. *medical journal armed forces india*, 77(3), pp.266-275.
- [28] Kavitha, S., Varuna, S. and Ramya, R., 2016, November. A comparative analysis on linear regression and support vector regression. In *2016 online international conference on green engineering and technologies (IC-GET)* (pp. 1-5). IEEE.
- [29] Smola, A.J. and Schölkopf, B., 2004. A tutorial on support vector regression. *Statistics and computing*, 14, pp.199-222.
- [30] Roy, A., Manna, R. and Chakraborty, S., 2019. Support vector regression based metamodeling for structural reliability analysis. *Probabilistic Engineering Mechanics*, 55, pp.78-89.
- [31] Columbia, 2024. Least Absolute Shrinkage and Selection Operator (LASSO). Available at: <https://www.publichealth.columbia.edu/research/population-health-methods/least-absolute-shrinkage-and-selection-operator-lasso> (Accessed 25.2.2024).
- [32] He, Y. and Wang, Y., 2021. Short-term wind power prediction based on EEMD–LASSO–QRNN model. *Applied Soft Computing*, 105, p.107288.
- [33] Hastie, T., Tibshirani, R., Friedman, J.H. and Friedman, J.H., 2009. *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer.
- [34] Fayyad, U.M. and Irani, K.B., 1992. On the handling of continuous-valued attributes in decision tree generation. *Machine learning*, 8, pp.87-102.
- [35] Scikit learn, 2024. Decision Trees. Available at: <https://scikit-learn.org/stable/modules/tree.html> (Accessed 25.2.2024).
- [36] Quinlan, J.R., 1986. Induction of decision trees. *Machine learning*, 1, pp.81-106.
- [37] Gulati, P., Sharma, A. and Gupta, M., 2016. Theoretical study of decision tree algorithms to identify pivotal factors for performance improvement: A review. *International Journal of Computer Applications*, 141(14), pp.19-25.
- [38] Hssina, B., Merbouha, A., Ezzikouri, H. and Erritali, M., 2014. A comparative study of decision tree ID3 and C4. 5. *International Journal of Advanced Computer Science and Applications*, 4(2), pp.13-19.
- [39] Song, Y.Y. and Ying, L.U., 2015. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2), p.130.

- [40] Loh, W.Y., 2011. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1), pp.14-23.
- [41] Steinberg, D. and Colla, P., 2009. CART: classification and regression trees. *The top ten algorithms in data mining*, 9, p.179.
- [42] Buhmann, M.D., 2000. Radial basis functions. *Acta numerica*, 9, pp.1-38.
- [43] de Cos, J., Sanchez, F., Ortega, F., Montequin, V., 2008. Rapid cost estimation of metallic components for the aerospace industry. *International Journal of Production Economics*, 112(1), 470-482.
- [44] Tran, V.T. and Yang, B.S., 2009. Data-driven approach to machine condition prognosis using least square regression tree. *Journal of mechanical science and technology*, 23, pp.1468-1475.
- [45] Torgo, L.F.R.A., 1999. Inductive learning of tree-based regression models.
- [46] Segal, M.R., 1988. Regression trees for censored data. *Biometrics*, pp.35-47.
- [47] Segal, M.R., 2004. Machine learning benchmarks and random forest regression.
- [48] Singh, B., Sihag, P. and Singh, K., 2017. Modelling of impact of water quality on infiltration rate of soil by random forest regression. *Modeling Earth Systems and Environment*, 3, pp.999-1004.
- [49] Breiman, L., 2001. Random forests. *Machine learning*, 45, pp.5-32.
- [50] Machine Learning, 2023. Random forests. Available at: <https://developers.google.com/machine-learning/decision-forests/random-forests> (Accessed 25.2.2024).
- [51] Medium, 2014. Analyzing Titanic Data With Random Forest In R. Available: <https://medium.com/edureka/random-forest-classifier-92123fd2b5f9> (Accessed 25.2.2024).
- [52] Github, 2024. ResidentMario / missingno. Available: <https://github.com/ResidentMario/missingno> (Accessed 25.2.2024).
- [53] Bookdown. 5.4 Data cleaning and imputation. Available: https://bookdown.org/martin_shepperd/ModernDataBook/C5-Cleaning.html (Accessed 25.2.2024).
- [54] Jäger, S., Allhorn, A., and Bießmann, F., 2021. A benchmark for data imputation methods. *Frontiers in big Data*, 4, 693674.
- [55] Karamlou, A., Bair, A. and Sharma, A., Imputation and Supervised Learning on Sparse Datasets.

- [56] Verdonck, T., Baesens, B., Óskarsdóttir, M. and vanden Broucke, S., 2021. Special issue on feature engineering editorial. *Machine Learning*, pp.1-12.
- [57] Machine Learning, 2022. Normalization Techniques at a Glance. Available: <https://developers.google.com/machine-learning/data-prep/transform/normalization> (Accessed 25.2.2024).
- [58] Cerda, P., Varoquaux, G., Kégl, B., 2018. Similarity encoding for learning with dirty categorical variables. *Machine Learning*, 107(8-10), 1477-1494.
- [59] Machine Learning, 2022. Transforming Categorical Data. Available: <https://developers.google.com/machine-learning/data-prep/transform/transform-categorical> (Accessed 25.2.2024).
- [60] Pargent, F., Pfisterer, F., Thomas, J. and Bischl, B., 2022. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. *Computational Statistics*, 37(5), pp.2671-2692.
- [61] Hancock, J.T. and Khoshgoftaar, T.M., 2020. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1), pp.1-41.
- [62] Idreos, S., Papaemmanouil, O. and Chaudhuri, S., 2015, May. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data* (pp. 277-281).
- [63] Zuur, A.F., Ieno, E.N. and Elphick, C.S., 2010. A protocol for data exploration to avoid common statistical problems. *Methods in ecology and evolution*, 1(1), pp.3-14.
- [64] Sedgwick, P., 2012. Pearson's correlation coefficient. *Bmj*, 345.
- [65] Schober, P., Boer, C. and Schwarte, L.A., 2018. Correlation coefficients: appropriate use and interpretation. *Anesthesia & analgesia*, 126(5), pp.1763-1768.
- [66] Chandrashekar, G. and Sahin, F., 2014. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1), pp.16-28.
- [67] Guyon, I. and Elisseeff, A., 2003. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar), pp.1157-1182.
- [68] Tan, J., Yang, J., Wu, S., Chen, G. and Zhao, J., 2021. A critical look at the current train/test split in machine learning. *arXiv preprint arXiv:2106.04525*.
- [69] Bertsimas, D. and Paskov, I., 2020. Stable regression: On the power of optimization over randomization in training regression problems. *The Journal of Machine Learning Research*, 21(1), pp.9374-9398.
- [70] Jung, Y., 2018. Multiple predicting K-fold cross-validation for model selection. *Journal of Nonparametric Statistics*, 30(1), pp.197-215.

- [71] Wikipedia, 2024. Coefficient of determination. Available: https://en.wikipedia.org/wiki/Coefficient_of_determination (Accessed 25.2.2024).
- [72] Scikit learn, 2024. sklearn.inspection.permutation_importance. Available: https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html (Accessed 25.2.2024).
- [73] Hancock, John T., and Taghi M. Khoshgoftaar, 2020. CatBoost for big data: an interdisciplinary review. *Journal of big data* 7(1), 94.
- [74] Altair, 2024. RapidMiner. Available: <https://altair.com/altair-rapidminer> (Accessed 10.3.2024).

APPENDIX A: OVERVIEW OF NUMERICAL DATA

	LABELLED_PRICE	ORDERED_PCS_12_MONTHS	SCALE_PRICE_EUR_PER_PC	SCALE_QUANTITY	WEIGHT_NETTO	SHOULD_COST	B	D2	E1	E2	H	L1	L2	OD	OD1	OD2	S	W
count	716	716	716	716	716	716	716	716	716	716	716	716	716	716	716	716	716	716
mean	15297758	3243754	0.502723	0.387285	0.738443	0.053073	4347451	1.006075	1211299	111662	6671927	701669	46.06816	23.2567	4.18575	4.47696	2.501536	22.2654
std	23.52821	268.679183	3.356464	2.793827	1.25891	1.420127	4771254	2.913645	5.245533	4.138548	57.00203	33.74214	28.74563	24.6095	8.96284	9.43364	0.983862	8.21916
min	0.01	0	0	0	0	0	0	0	0	0	0	18	9	0	0	0	0	0
25%	556	0	0	0	0.12825	0	16	8.5	9375	10	30.5	50	26	4.5	0	0	2	20
50%	11.135	0	0	0	0.31	0	28	10.2	11	10	48	62	38	18	0	0	3	20
75%	18.3825	7.25	0	0	0.8225	0	52.25	11	14.5	12.5	83	82	60	35.5	0	0	3	25
max	426.81	6650	45	50	13.33	38	500	32	48.5	60	546	326	265	219	60.5	60.5	10	120