Innovating Sales and Planning of Complex Industrial Products
Exploiting Artificial Intelligence

# Deliverable 3.5
# Optimal Pricing Component

| | |
|---|---|
| Deliverable type: | Software |
| Deliverable reference number: | ITEA 20054 \| D3.5 |
| Related Work Package: | WP 3 |
| Due date: | 2024-09-30 |
| Actual submission date: | 2025-01-15 |
| Responsible organisation: | IFAK |
| Editor: | Mario Thron |
| Dissemination level: | Public |
| Revision: | Final \| Version 1.0 |

| | |
|---|---|
| Abstract: | This document describes the design, development, and deployment specifications for the Optimal Pricing Component. It summarizes methodologies from artificial neural networks, statistical forecasting, and fuzzy logic-based models derived from Task 3.5 to improve pricing strategies. |
| Keywords: | optimized pricing, artificial neural networks, statistics and forecasting, fuzzy logic |

| Table_head | TUD | EM: | Approval date (1 / 2) |
|---|---|---|---|
| Approval at WP level | OK | OK | 2025-12-12 |
| Veto Review by All Partners | | | 2025-01-15 |

**Editor**

Mario Thron (IFAK)

**Contributors**

Mario Thron (IFAK)

Thomas Marus (Demag)

Christian Mai (Demag)

Frank Werner (Software AG)

Mahmoud Mohamed (Software AG)

Ahmet Emin Ünal (Adesso)

## Executive Summary

This report introduces a set of innovative technologies for pricing, including 3D Shape-based pricing strategies, AI-driven forecasting systems, and fuzzy logic-based pricing methods. These approaches enhance pricing strategies by reducing manual intervention and improving responsiveness. Here is a short summary of the results:

- **3D Shape-based Pricing Strategies**: Chapter 2 presents an innovative method using advanced machine learning models to process 3D data for cost estimation in industries like metal sheet stamping and mold-making. By integrating voxelization techniques and other geometric analyses, the approach predicts labour costs based on part geometry, material properties, and production conditions. This results in a significant reduction of mean absolute percentage error (MAPE) from traditional methods to around 10% using LightGBM and XGBoost models. The integration of these strategies not only enhances cost prediction accuracy but also accelerates the quotation process.
- **Pricing Strategies Based on Statistics and Forecasting**: Chapter 3 introduces a novel AI-based system that automates master price list adjustments using statistical analysis and forecasting. The system integrates real-time data to generate accurate forecasts, enabling swift pricing adjustments based on dynamic market conditions. NeuralProphet outperforms traditional models in capturing complex patterns, particularly for aluminium prices. Real-time transparency fosters trust and customer satisfaction by providing clear rationales for pricing decisions. Data privacy is preserved using index prices, ensuring sensitive company information remains secure.
- **Fuzzy Logic-Based Pricing Strategies**: Chapter 4 explores fuzzy logic-based pricing strategies at the Bill of Material (BOM) level. The Fuzzy Control Language Engine (FCLE), adhering to IEC 61131-7 standards, automates price adjustments based on factors like factory load, market demand, and production costs. The FCLE offers a comprehensive RESTful API for seamless integration into existing systems. Its modular design includes components such as engine.py, parser.py, service.py, and io_connectors.py. Dynamic adaptability through fuzzy logic rules reduces manual intervention by sales engineers, while Mamdani’s inference method is preferred for handling multiple output variables.

Future efforts will focus on integrating external data acquisition capabilities and conducting extensive testing at demonstrator sites to refine the system's performance under real-world conditions.

## Table of Content

## Figures

## Tables

**No table of figures entries found.**

# 1 Introduction

The InnoSale research project aims to develop advanced pricing strategies by leveraging modern technologies such as statistical forecasting and fuzzy logic-based systems. This report presents key approaches developed during the project: a novel pricing approach based on 3D shapes of sheet metal plates, a novel AI-based system for dynamic pricing adjustments and a Fuzzy Control Language Engine (FCLE) designed for robust and flexible price management.

In Section 2, we explore a revolutionary method for cost estimation using advanced machine learning models to process 3D data. This approach is particularly useful in industries like metal sheet stamping and mold-making, where accurate labour cost predictions are crucial. By integrating voxelization techniques and other geometric analyses, the system reduces mean absolute percentage error (MAPE) from traditional methods to around 10% using LightGBM and XGBoost models. The integration of these strategies not only enhances cost prediction accuracy but also accelerates the quotation process by reducing manual intervention and improving responsiveness.

In Section 3, we introduce an innovative AI-driven system that automates master price list adjustments using statistical analysis and forecasting. By integrating real-time data from various sources, the system generates accurate forecasts to enable swift pricing adjustments based on dynamic market conditions. This approach not only enhances accuracy but also reduces the need for manual intervention by product managers. Additionally, it fosters trust and customer satisfaction through transparent and clear rationales for pricing decisions while preserving sensitive company information using index prices.

Section 4 focuses on fuzzy logic-based pricing strategies at the Bill of Material (BOM) level. The FCLE, which adheres to IEC 61131-7 standards, automates price adjustments based on factors such as factory load, market demand, and production costs. With a comprehensive RESTful API for seamless integration into existing systems and a modular design, the FCLE reduces manual intervention by sales engineers and improves efficiency through dynamic adaptability using fuzzy logic rules.

The report also indicates plans for future work, including integrating external data acquisition capabilities and conducting extensive testing at demonstrator sites to further refine system performance under real-world conditions.

# 2 3D Shape-based Pricing Strategies

This chapter delves into an innovative approach for pricing strategies, driven by 3D shape-based data analysis. Using advanced machine learning models tailored to process 3D data, this method is designed to predict manufacturing costs, primarily in industries where custom parts are produced, such as the metal sheet stamping and mold-making sectors. By leveraging voxelization and other 3D geometric techniques, companies can enhance their cost estimation processes, reducing uncertainties and enabling more accurate pricing.

## 2.1 Motivation

The growing complexity of custom part production, particularly in industries like metal sheet stamping, has necessitated a shift toward more advanced pricing strategies. Traditional pricing methods, which rely heavily on manual cost estimation and historical data, often struggle to keep up with the dynamic nature of short-run, customized production. This challenge is especially evident in sectors where factors such as material properties, part geometry, and the production operations are key determinants of the cost.

For manufacturers like those in the metal sheet stamping industry, accurately predicting the cost of custom parts is essential not only for ensuring profitability but also for maintaining competitiveness in the market. Clients demand quick and accurate quotations, especially when custom configurations and intricate designs are involved. The variability in design, materials, and production processes in these industries makes traditional pricing models inefficient. This is where the integration of 3D shape data into pricing strategies becomes critical.

For manufacturers, while accurately predicting the cost of custom parts, it is crucial to adhere to the customer contracts. These contracts often define fixed input (time independent) parameters, such as the number of operations, part geometry, material type (e.g., steel or alloys), and material thickness. Given that these factors remain constant over time, they provide a stable foundation for long-term production planning. However, material property considerations such as hardness, strength, lightness, and elongation—combined with fluctuating steel prices (as reflected in indices such as the Steel Price Index or the Ductile Iron Castings index)—introduce additional layers of complexity in cost estimation. As customer contracts fix these parameters over extended periods, manufacturers must ensure their pricing models can efficiently account for such fixed inputs.

On the other hand, variable input parameters, such as labour costs, worker proficiency, investment in new technology, production capacity, and scrap factors, fluctuate more frequently and can significantly affect production costs. Labor-related variables, including hourly wages and worker proficiency, can greatly influence both operational efficiency and total production cost. Additionally, confidential factors like labour workload need to be carefully handled and integrated into the pricing strategy to ensure competitive and accurate cost predictions without compromising sensitive information.

In such a context, the limitations of traditional cost estimation methods become increasingly apparent. Manual cost estimation, which depends on expert knowledge and historical data, often fails to provide the agility needed to adjust for the dynamic aspects of custom production. Moreover, the time-intensive nature of gathering and analysing data from various

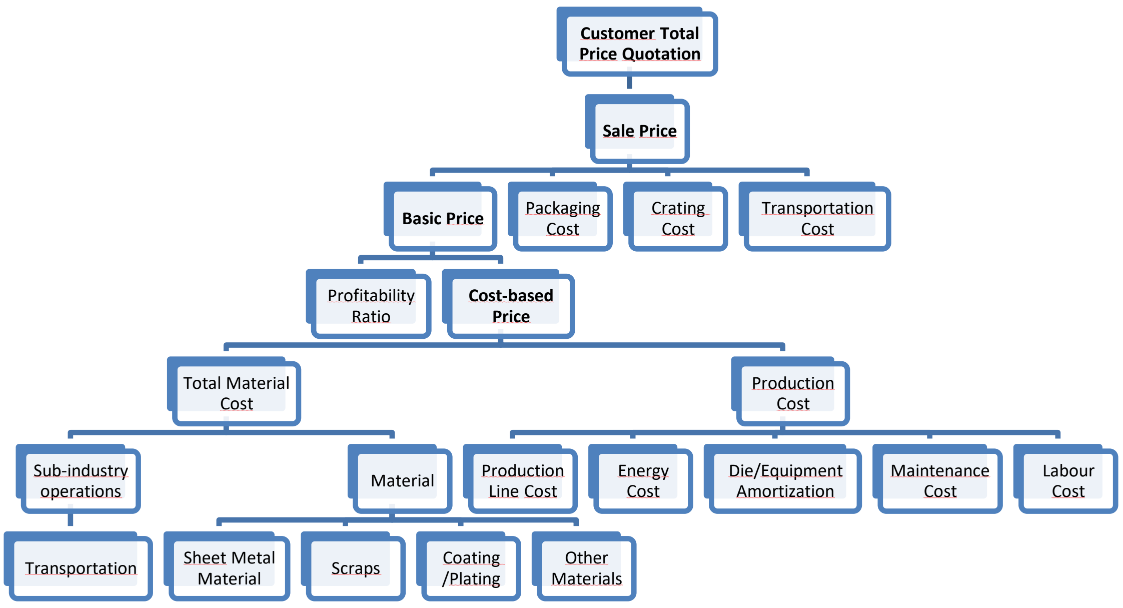


**Figure 1. Breakdown of the costs used for price calculation**

sources further slows down the quotation process, limiting a company's ability to respond quickly to customer inquiries. As an example, analysing a 3D part and predicting the labour cost of a part quotation can take from 10 minutes to 30 minutes in average, which can accumulate into very large delays for large projects.

By adopting a 3D shape-based pricing strategy, manufacturers can automate the cost estimation process, significantly improving efficiency and accuracy by reducing human error. Advanced machine learning models that process 3D data—integrated with material properties, incorporating fixed input parameters, and predicting the variable factors—offer a comprehensive solution. Such models can capture complex interdependencies between geometric features, material properties, and production conditions, enabling manufacturers to provide more accurate, and real-time dynamic pricing. This approach is particularly beneficial for addressing the challenges posed by varying product dimensions, complex geometries, and custom mold designs, which have a significant impact on both material and labour costs.

Ultimately, this approach allows companies to ensure their pricing strategies remain competitive and transparent, even in the face of fluctuating labour costs, material prices, and operational efficiencies. For industries reliant on customer contracts with fixed parameters, this dynamic, data-driven method offers the flexibility needed to optimize both short-term production costs and long-term pricing models.

## 2.2 Principles

The primary goal of 3D shape-based pricing strategies is to integrate advanced machine learning algorithms with 3D shape recognition techniques to improve cost prediction accuracy. The main prediction focus was the labour costs, that makes up the largest part of the variable input parameters.

Accurately predicting labour costs in custom manufacturing is heavily influenced by the complexity of the 3D part data, as each part's geometry, size, and design intricacies directly impact the required number of operations and worker proficiency needed to complete the task. Analysing the 3D part data enables manufacturers to estimate the specific time and effort required for each production step, considering factors like intricate contours, multiple surface finishes, or unique shapes that may demand specialized skills or extended labour hours. By integrating 3D data into labour cost prediction, companies can more precisely forecast workforce allocation, streamline scheduling, and optimize cost efficiency, resulting in more accurate and competitive pricing for custom parts.

The methodology revolves around the following key principles:

- **Voxelization for 3D Feature Extraction**: Voxelization is a core process where 3D models, such as those used in metal stamping or mold-making, are converted into structured grid data (voxels). These voxels represent the geometric features of the product, enabling the machine learning models to process complex shapes and predict related costs based on their structure. In our study, we sampled points tangent to the part surface to generate a point cloud, which then used to generate the voxelized part by quantization of the point locations to a configured resolution (Fig. 2). Voxelization of a 3D model allows for the effective application of convolutional neural networks (CNNs) and/or other image-based models, which can recognize intricate patterns in 3D shapes. For our case we used this voxelization approach to calculate the volume of the sheet metal part.

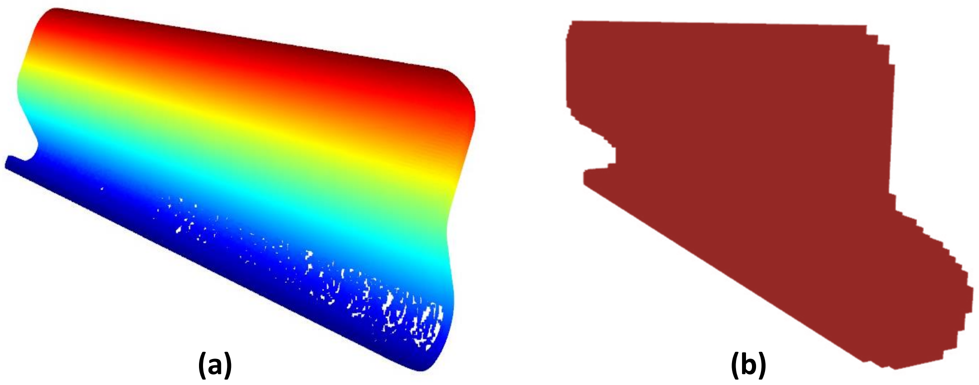


**Figure 2. (a) Point cloud sampling on the original part. (b) Generated voxelized part.**

We also used the mesh of the part to calculate the surface area and triangle count, which is indirectly correlated with the complexity of the part production.

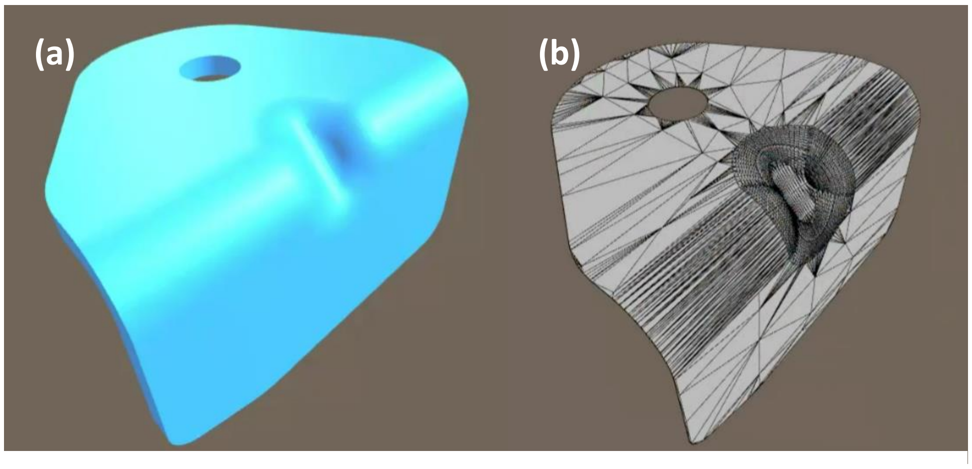


**Figure 3. (a) Original part. (b) Mesh of the part.**

- **Integration of Product and Process Data**: The models are trained on a combination of 3D shape features and other critical factors of parts such as material types (which affects tensile strength, sheet elongation, metal hardness), sheet specific information (such as sheet thickness and net dimensions), and part information (such as part contour size and surface area). Each part also includes information related to the sequential operations that needs to be applied to the metal sheet to produce the required part. The operation information such as press tonnage, mold information (such as dimensions, weight, and fill), operation directions and sub operations (such as blanking, shearing, bending, etc.) and the operation labour cost (work man-hour). This ensures that the machine learning models not only consider the geometric complexity of the product but also the production conditions. This integrated approach helps predict not just the material cost but also the man-hour cost required for each operation.

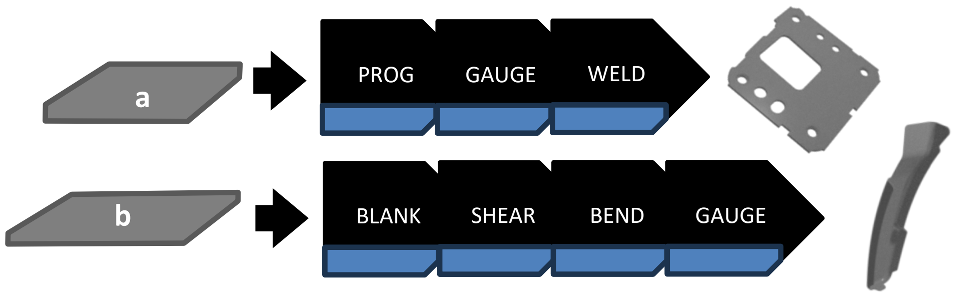


**Figure 4. Representation of the dataset structure.**

- **Predictive Modeling Using Gradient Boosting Algorithms**: In this study, gradient boosting algorithms such as LightGBM and XGBoost are employed due to their strong

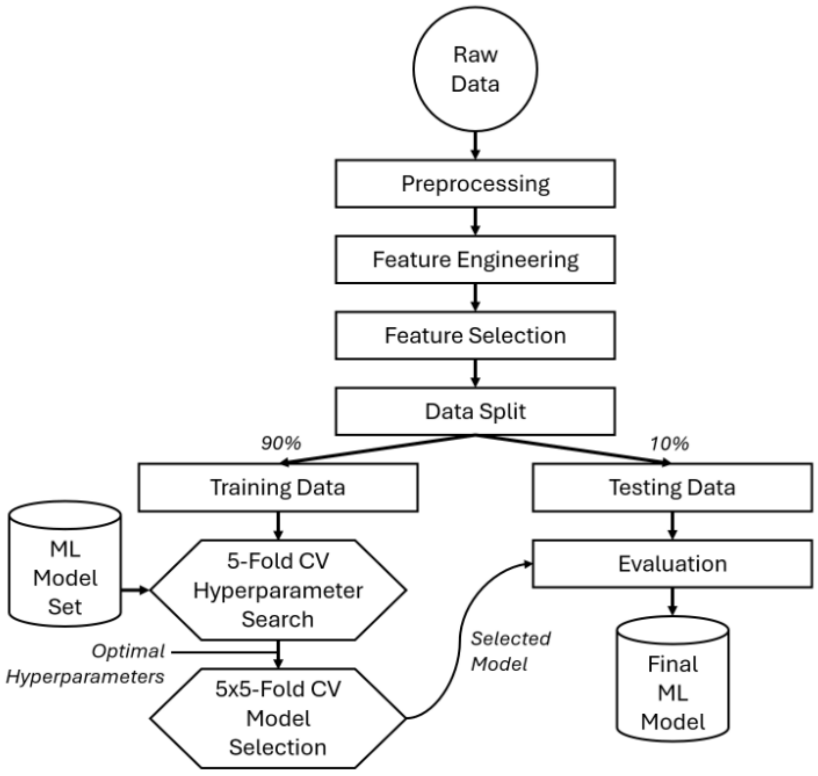


**Figure 5. The flow diagram of the process.**

performance in handling tabular data and their ability to model complex non-linear relationships. These models are enhanced through the incorporation of domain-specific knowledge—such as the sequence of operations in metal stamping—into the feature engineering process.

- **Synthetic Data Generation for Feature Enrichment**: Given the relatively limited availability of real-world data in some manufacturing sectors, synthetic data generation is used to augment the dataset. By simulating variations in product dimensions, materials, and other key attributes, the model's robustness is improved. This also helps in addressing the lack of sufficient data for certain edge cases. For this reason, we used SMOTE (Synthetic Minority Oversampling Technique) algorithm to generate new data points like the instances of the datasets, and then used Tomek Link algorithm to under sample and cleanup the overlapping instances.
- **Explainable AI with SHAP Analysis**: To ensure transparency and trust in the predictions, SHapley Additive exPlanations (SHAP) are used to interpret the contribution of each feature to the final cost prediction. This enables the identification of the most influential factors, such as mold size, press tonnage, or specific geometric features, thus offering insights into why certain predictions are made.

The integration of these principles into the pricing strategy allows manufacturers to achieve greater precision in their cost estimates, which is especially crucial in the competitive landscape of custom manufacturing.

## 2.3 Results

The implementation of 3D shape-based pricing strategies has yielded significant improvements in the accuracy of cost predictions across various case studies, particularly in the metal sheet stamping sector.

By integrating 3D shape data into the cost prediction models, manufacturers observed a reduction in mean absolute percentage error (MAPE) of 10% using the proposed LightGBM and XGBoost models. This value is within an acceptable range, as the predicted values using the traditional methods have similar confidence interval percentage. This improvement in accuracy is particularly evident in complex, custom part orders where traditional cost estimation methods often struggle.

**Table 1. Comparison table of the cross validation results**

| Models | Results | |
|---|---|---|
| | $5 \times 5$ *CV MAPE* | $5 \times 5$ *CV MAE* |
| LightGBM* | 10.89 | 71.49 |
| LightGBM | 10.78 | 70.37 |
| XgBoost* | 11.30 | 73.72 |
| XgBoost | 11.23 | 72.25 |
| KNN Regressor | 20.55 | 122.01 |
| MLP | 16.61 | 105.25 |
| Linear Regression | 26.35 | 136.99 |

** indicates that the model is trained without the additional 3D data features for comparison.*

The automation of cost estimation through machine learning models significantly reduced the time required to generate quotations. On average the inference time of the model is less than

15 seconds. However the analysis of the 3D part can take some time if not already stored in the database, which makes the total average inference around 2 minutes ± 15 seconds with the analysis included. Considering the reported time from the manufacturer, this is a reduction of 90% in the time taken to respond to customer inquiries, which is crucial in industries where speed is a competitive advantage.

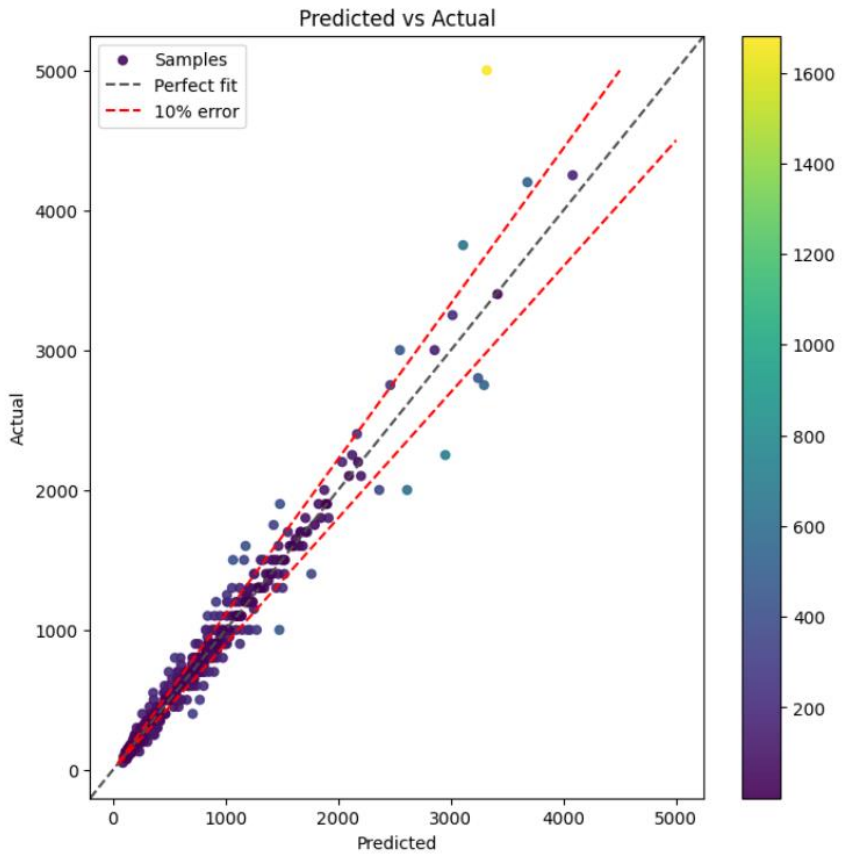


**Figure 6. Evaluation of the best model on the test set.**

The use of machine learning models allowed for scalability, as the models can easily be retrained with new data and adapted to different product categories. This flexibility is vital for manufacturers that handle a wide variety of custom orders, where each project may require different materials, processes, and operations. By providing more transparent pricing models

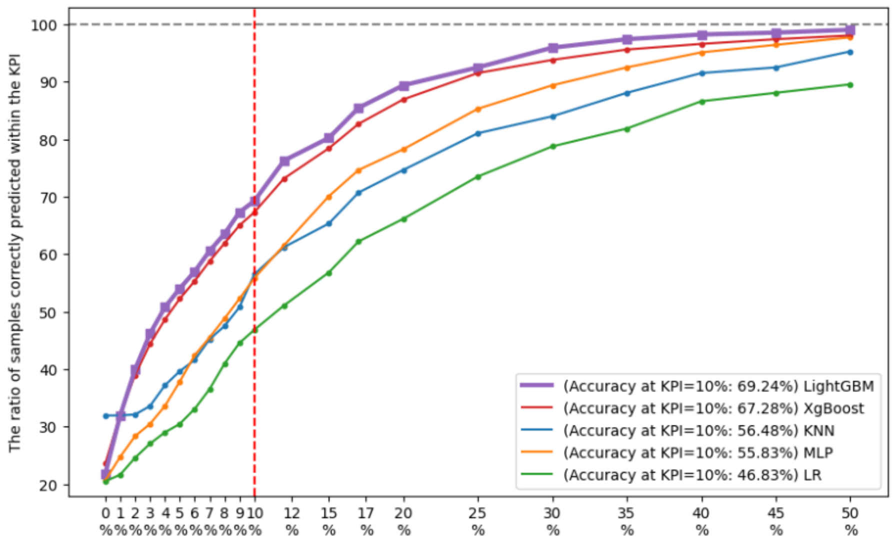


**Figure 7. Change of the ratio of samples correctly predicted within different KPI**

based on data-driven predictions, companies were able to offer clearer justifications for their pricing. This increased transparency has been shown to improve customer satisfaction and trust, as clients are given more detailed insights into the cost drivers of their custom products.

SHAP analysis revealed that certain geometric features, such as mold volume and the presence of specific operations (e.g., progressive stamping), have a significant impact on cost predictions. This insight allowed manufacturers to focus on optimizing these key parameters, further reducing production costs.

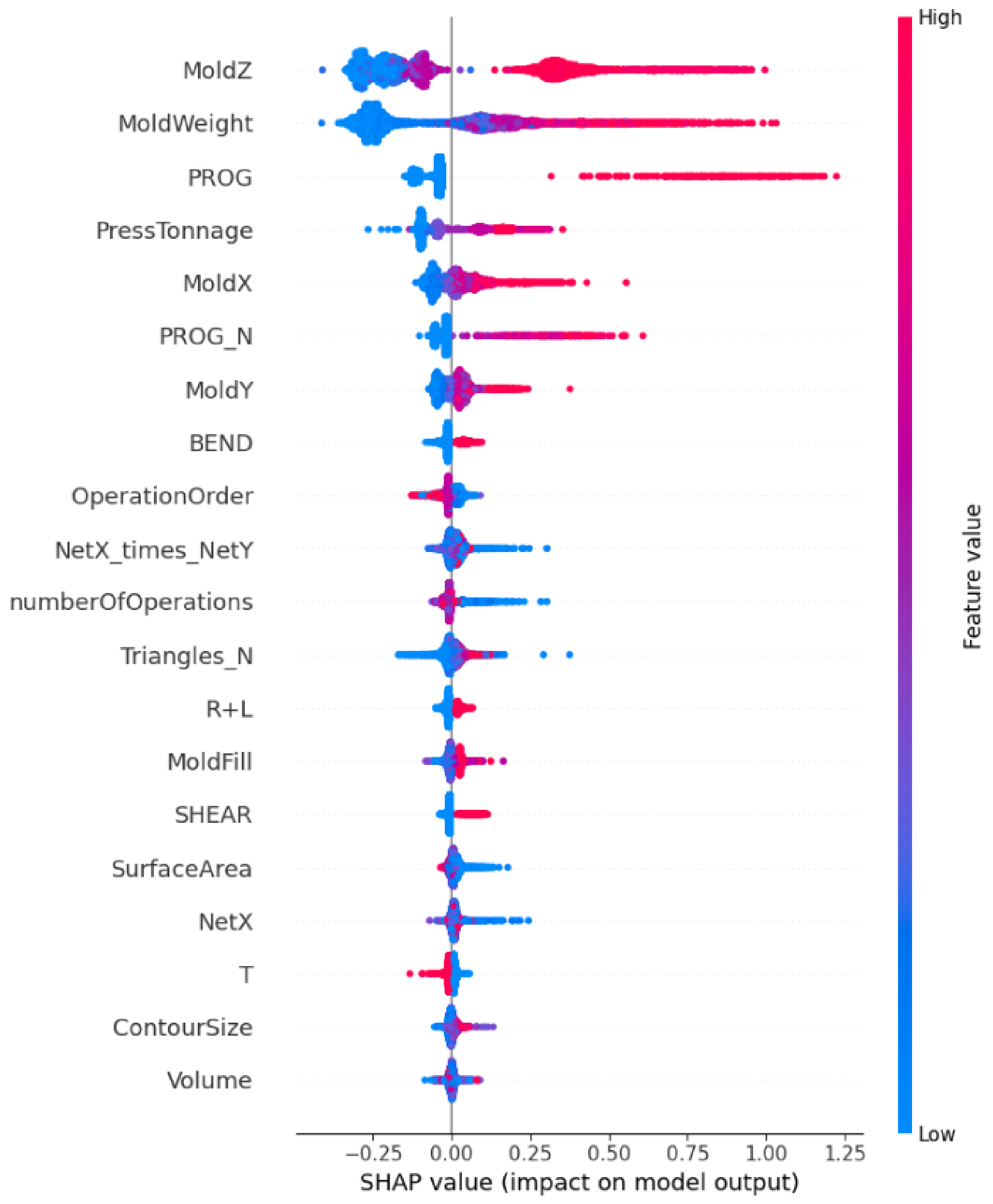


**Figure 8. The SHAPley summarization of the features.**

The service is deployed as an API into on-premises and tested, more technical detail regarding the documentation and API usage is reported under the 5.1.4 heading of the D3.7 document.

The results demonstrate that 3D shape-based pricing strategies can significantly enhance cost prediction models, providing manufacturers with a competitive edge. The inclusion of 3D shape data not only improves accuracy but also offers greater flexibility and speed in the pricing process, making it an essential tool in the era of Industry 4.0.

# 3 Pricing Strategies based on Statistics and Forecasting

## 3.1 Motivation

This chapter addresses innovative pricing strategies using artificial intelligence based on statistical analysis and forecasting. In the current workflow for adjusting the master price list, the prices must be defined by the product managers based on the available material, production and engineering costs as well as the target margin. This process is extremely time intensive and inflexible because all the information needs to be found manually from different sources and then implement these adjustments into the master price list. Moreover, a short-term anticipation of material cost development is difficult to achieve. Also, material cost forecast has less precision for long term forecast compared to short term forecast. This causes difficulties for defining the suitable adjustment rate. The AI-based approach pursues the automated adjustment of the master price list for the different product groups in light lifting equipment. Internal data and parameters for each material item are thoroughly analysed in advance and a forecast is created with the help of external data sources. Material consumption, production times, current and future purchase prices from suppliers, stock levels, production and engineering costs as well as logistical costs should be considered. Of particular interest is the cost development of certain volatile materials such as steel, copper and aluminium.

Real-time access to price developments enables an automated process for generating proposals for price adjustments or material surcharges. This approach considers key factors within the material price list and creates the basis for optimal pricing. The efficiency of the entire process is increased as less manual work is required. The improved quality of the forecasts leads to higher price quality and therefore stable margins.

The primary objective of developing AI based forecasting service was to develop accurate forecasting models that can reliably predict the future prices of materials we are concerned with in production, aiding in effective decision-making and resource allocation for the production process. Leveraging advanced analytical techniques and machine learning algorithms, we aimed to identify key trends, patterns, and correlations within the dataset to create robust predictive models, by considering various external economic factors such as material prices, interest rates, and inflation, the goal is to provide an insight of potential price movements in the future.

## 3.2 Data Collection

The data utilized in this report has been gathered from a combination of trustful sources, ensuring a comprehensive and reliable data set for our analysis. The next figure shows a preview of our data hierarchy.

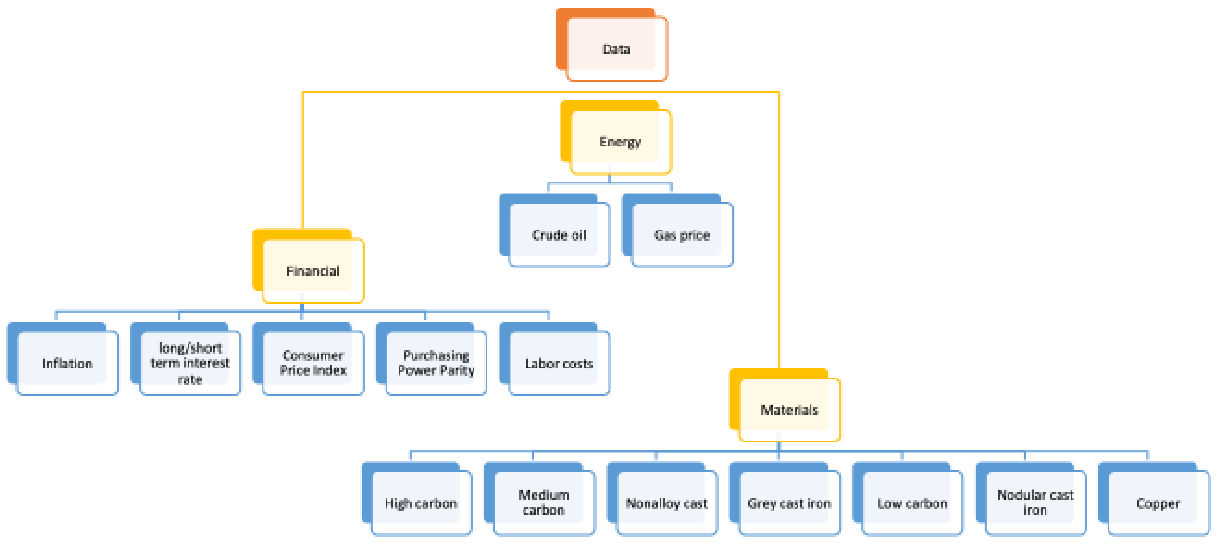


**Figure 9: Input Sources and categories used for model building**

### 3.2.1 Material price trends

The material price trends are gathered from Federal Reserve Economic Data[1].

The product we are concerned with is essentially produced from these six main alloys:

**(High carbon - medium carbon - cast non-alloy and low alloy steels - grey cast iron – nodular cast iron - low carbon steel)**

Aluminium and copper are critical components in a wide range of products, so we made it a priority to include comprehensive data on both materials. To ensure accuracy and reliability, we sourced copper data (oberen Kupfer-DEL-Notiz) from Westmetall[2] and aluminium data from IndexMundi [3].

### 3.2.2 Financial Price Trends

Financial assets and economic indicators such as the inflation rate, short-term and long-term interest rates, purchasing power parity (PPP), and labour costs play a crucial role in determining production costs. When the inflation rate rises, the general price level in an economy increases, leading to higher costs for raw materials, energy, and other inputs essential for production. This makes it more expensive for companies to produce goods and services. Interest rates, both short-term and long-term, also affect production costs. Higher interest rates increase the cost of borrowing for businesses, raising the expenses associated with financing operations, purchasing equipment, or investing in new projects. Conversely, lower interest rates can reduce these costs, making production more affordable. Purchasing power parity (PPP) impacts production costs by reflecting the relative value of currencies between countries. If a country's currency weakens, imported goods and services become more expensive, driving up the cost of inputs for production that rely on foreign resources.

[1] https://fred.stlouisfed.org/

[2] https://www.westmetall.com/en/markdaten.php?action=averages&field=DEL_high

[3] https://www.indexmundi.com/commodities/?commodity=aluminum&months=180&currency=eur

Labor costs are another significant factor. When wages and benefits rise, the cost of labour increases, directly raising the overall cost of production. Conversely, lower labour costs can reduce production expenses, although this can sometimes come at the expense of productivity or quality.

Together, these financial factors create a complex environment that directly influences how much it costs to produce goods and services, affecting pricing, profitability, and competitiveness in the market. The financial historical we are interested in are more of big-picture economic factors, so we chose those five economical historical data.

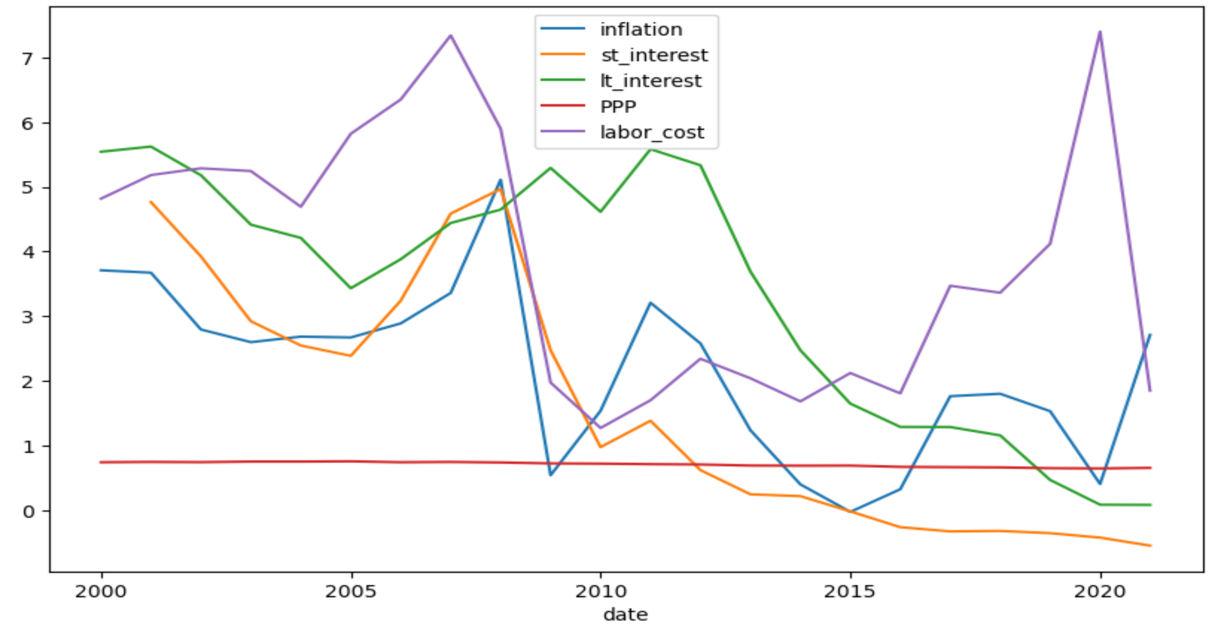


**Figure 10: Plot showing series of PPP, inflation, short-term interest rate, long-term interest rate, and labour costs**

### 3.2.3 Energy data

The production of crude metals and alloys is a highly energy-intensive process, relying heavily on large amounts of oil, coal, and natural gas. From the initial extraction of metal ores through mining to the complex processes of smelting and refining, every step requires significant energy input. Heavy machinery used in mining operations is typically powered by diesel, a byproduct of oil refining. Once the ore is extracted, it must be crushed, ground, and processed to extract the metal, all of which demand considerable amounts of energy. Smelting, where metal is extracted from its ore at high temperatures, is particularly reliant on coal, oil, and natural gas to generate the necessary heat. This reliance on fossil fuels not only drives up energy consumption but also has significant environmental impacts, contributing to carbon emissions and resource depletion.

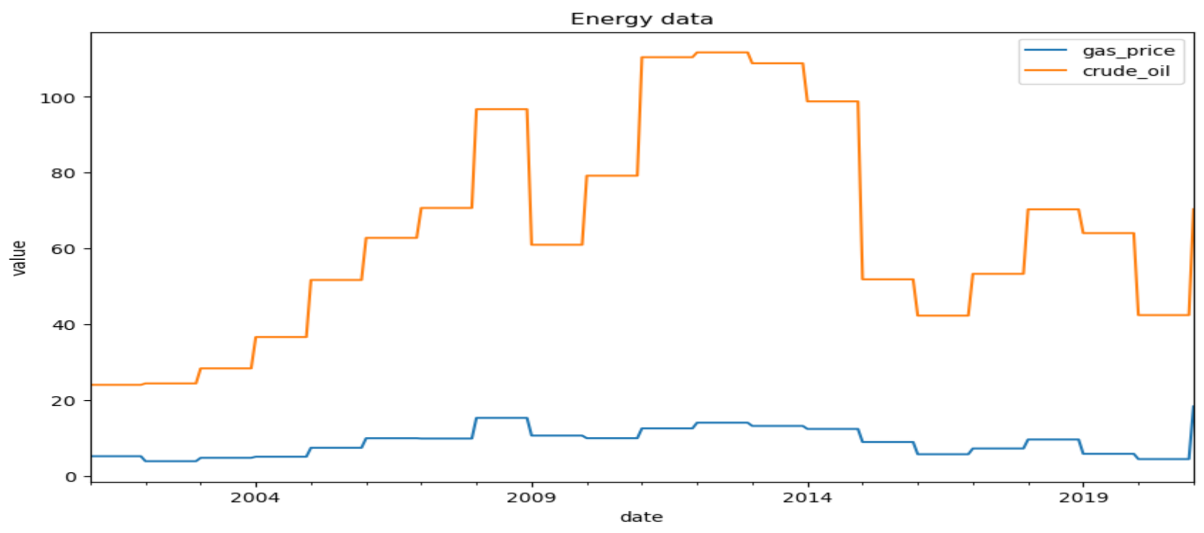


**Figure 11: Plot showing the data sets of gas and crude-oil price**

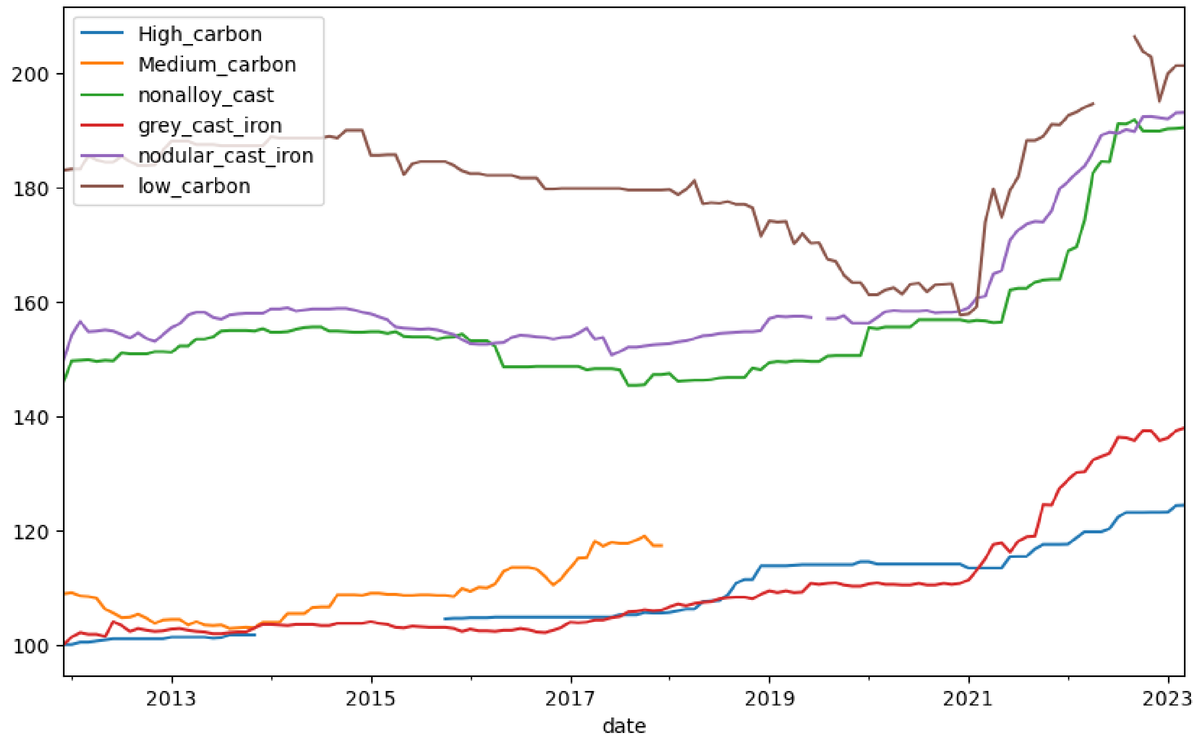
The previous plot reveals that, despite differences in their absolute values, gas and crude oil prices exhibit similar patterns over time. This correlation suggests that when incorporating energy factors into the model, including just one of these variables is sufficient to capture the overall trend, as adding both would likely introduce redundancy without providing significant additional insight.

## 3.3 Data Processing

First for some data we only have yearly captured values we need to convert them to monthly values, so we have consistent data frequency across all data frames. Some data also needed to be transformed from string data type to float so we have consistent type.

### 3.3.1 Healing Gaps in Data Series

After investigating the material price trend across time as you can see in this plot:

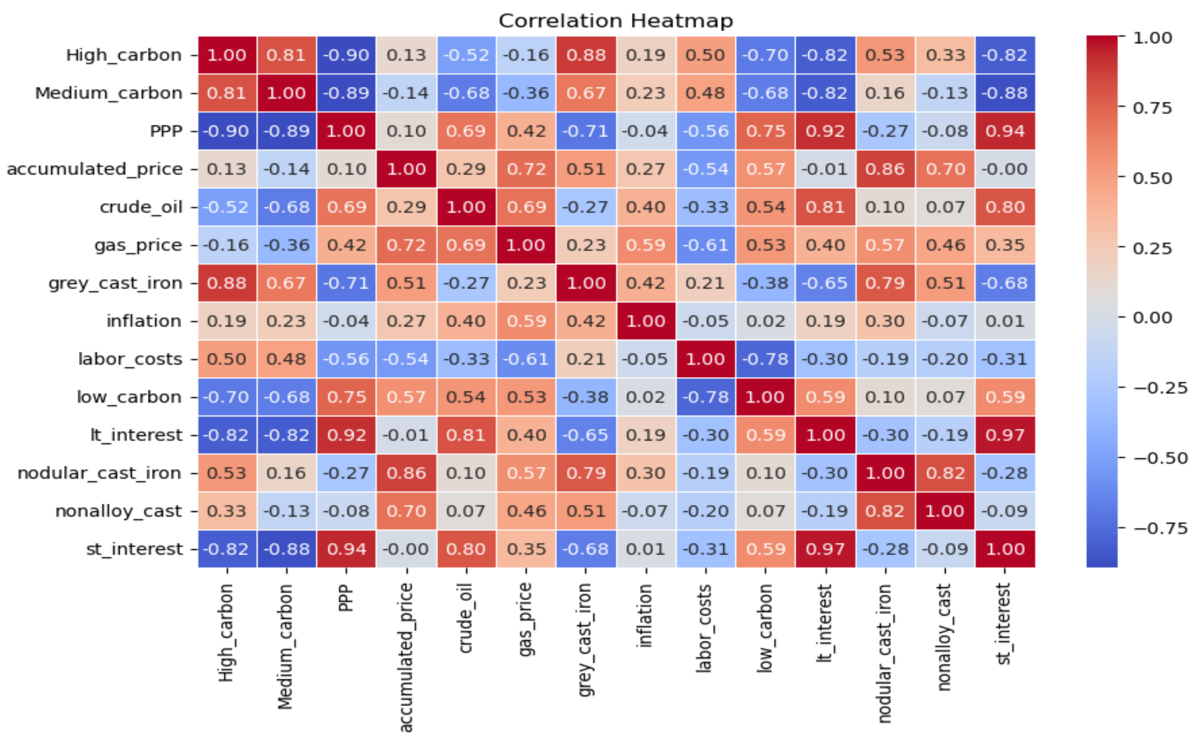


There were significant gaps in the data over certain periods, particularly for medium carbon, where the statistics were discontinued in 2018. To address this issue and maintain the integrity of our analysis, we chose to fill these gaps using linear interpolation.

### 3.3.2 Correlation map

A correlation matrix is a table that displays the correlation coefficients between different variables in a dataset. In the context of AI model training, this matrix is a valuable tool for understanding the relationships between features, or input variables, in your dataset. The correlation coefficient, often denoted by *r*, measures the strength and direction of the linear relationship between two variables. It ranges from -1 to 1, where *r = 1* indicates a perfect positive correlation, meaning as one variable increases, the other also increases proportionally. Conversely, *r = -1* indicates a perfect negative correlation, whereas one variable increases, the other decreases. A correlation of *r = 0* suggests no linear relationship between the variables.

Understanding these correlations helps in AI model training by identifying which features are strongly related and which might be redundant. Highly correlated features can sometimes lead to multicollinearity, which can negatively impact the model's performance by making it difficult to determine the individual effect of each feature. On the other hand, understanding low or negative correlations might highlight features that bring unique information to the model. Overall, the correlation matrix provides a foundational understanding of the data structure, guiding the selection and engineering of features to improve model accuracy and efficiency.



| | High_carbon | Medium_carbon | PPP | accumulated_price | crude_oil | gas_price | grey_cast_iron | inflation | labor_costs | low_carbon | lt_interest | nodular_cast_iron | nonalloy_cast | st_interest |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| High_carbon | 1.00 | 0.81 | -0.90 | 0.13 | -0.52 | -0.16 | 0.88 | 0.19 | 0.50 | -0.70 | -0.82 | 0.53 | 0.33 | -0.82 |
| Medium_carbon | 0.81 | 1.00 | -0.89 | -0.14 | -0.68 | -0.36 | 0.67 | 0.23 | 0.48 | -0.68 | -0.82 | 0.16 | -0.13 | -0.88 |
| PPP | -0.90 | -0.89 | 1.00 | 0.10 | 0.69 | 0.42 | -0.71 | -0.04 | -0.56 | 0.75 | 0.92 | -0.27 | -0.08 | 0.94 |
| accumulated_price | 0.13 | -0.14 | 0.10 | 1.00 | 0.29 | 0.72 | 0.51 | 0.27 | -0.54 | 0.57 | -0.01 | 0.86 | 0.70 | -0.00 |
| crude_oil | -0.52 | -0.68 | 0.69 | 0.29 | 1.00 | 0.69 | -0.27 | 0.40 | -0.33 | 0.54 | 0.81 | 0.10 | 0.07 | 0.80 |
| gas_price | -0.16 | -0.36 | 0.42 | 0.72 | 0.69 | 1.00 | 0.23 | 0.59 | -0.61 | 0.53 | 0.40 | 0.57 | 0.46 | 0.35 |
| grey_cast_iron | 0.88 | 0.67 | -0.71 | 0.51 | -0.27 | 0.23 | 1.00 | 0.42 | 0.21 | -0.38 | -0.65 | 0.79 | 0.51 | -0.68 |
| inflation | 0.19 | 0.23 | -0.04 | 0.27 | 0.40 | 0.59 | 0.42 | 1.00 | -0.05 | 0.02 | 0.19 | 0.30 | -0.07 | 0.01 |
| labor_costs | 0.50 | 0.48 | -0.56 | -0.54 | -0.33 | -0.61 | 0.21 | -0.05 | 1.00 | -0.78 | -0.30 | -0.19 | -0.20 | -0.31 |
| low_carbon | -0.70 | -0.68 | 0.75 | 0.57 | 0.54 | 0.53 | -0.38 | 0.02 | -0.78 | 1.00 | 0.59 | 0.10 | 0.07 | 0.59 |
| lt_interest | -0.82 | -0.82 | 0.92 | -0.01 | 0.81 | 0.40 | -0.65 | 0.19 | -0.30 | 0.59 | 1.00 | -0.30 | -0.19 | 0.97 |
| nodular_cast_iron | 0.53 | 0.16 | -0.27 | 0.86 | 0.10 | 0.57 | 0.79 | 0.30 | -0.19 | 0.10 | -0.30 | 1.00 | 0.82 | -0.28 |
| nonalloy_cast | 0.33 | -0.13 | -0.08 | 0.70 | 0.07 | 0.46 | 0.51 | -0.07 | -0.20 | 0.07 | -0.19 | 0.82 | 1.00 | -0.09 |
| st_interest | -0.82 | -0.88 | 0.94 | -0.00 | 0.80 | 0.35 | -0.68 | 0.01 | -0.31 | 0.59 | 0.97 | -0.28 | -0.09 | 1.00 |

## 3.4 SHAP Model

As our initial approach to the price prediction task, we employed the Prophet model. While it performed well for most alloys' price histories, it struggled significantly with aluminium, delivering poor results.

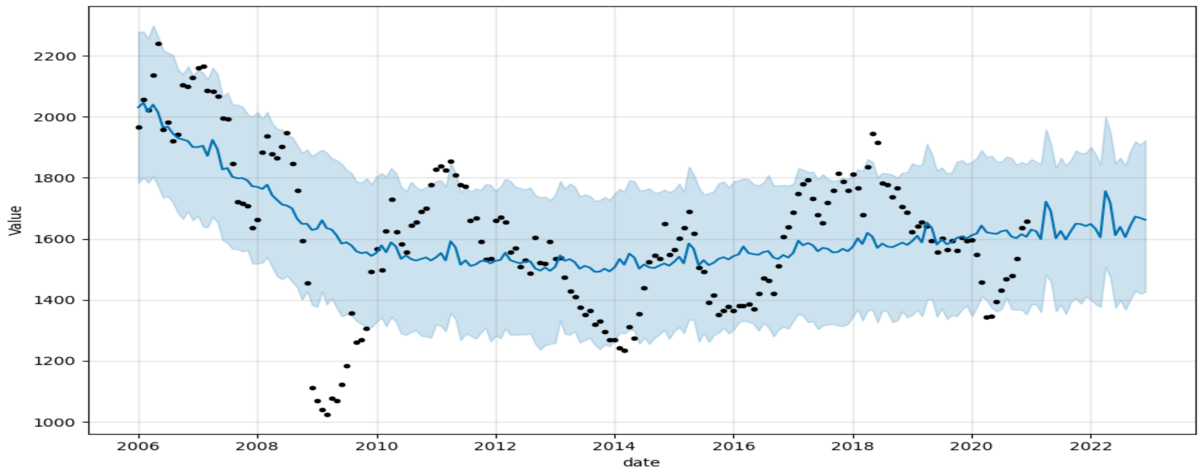


**Figure 12: AI Model results that show the mismatch between the actual data (black dots), and the trained model (blue line). The light-blue area indicates the confidence interval.**

To gain a deeper understanding of the factors influencing the final predicted prices, we conducted a SHAP (SHapley Additive exPlanations) analysis, which allowed us to thoroughly investigate the contribution of each variable to the model's predictions. SHAP is a method used in data analytics and AI model training to explain the output of machine learning models by assessing the contribution of each feature to the predictions. Derived from Shapley values in game theory, SHAP provides a way to interpret complex models, such as ensemble methods or deep learning models, which are often considered "black boxes." By calculating SHAP values, we can determine how much each feature contributes to or detracts from the prediction for a specific data point. This interpretability is crucial for understanding model behaviour, identifying important features, and ensuring transparency, especially in applications where understanding the decision-making process is essential. SHAP values offer a consistent and unified approach to feature importance, making it easier to compare different models and explain their predictions to non-technical stakeholders. Overall, SHAP enhances trust in AI models by making their predictions more transparent and interpretable.

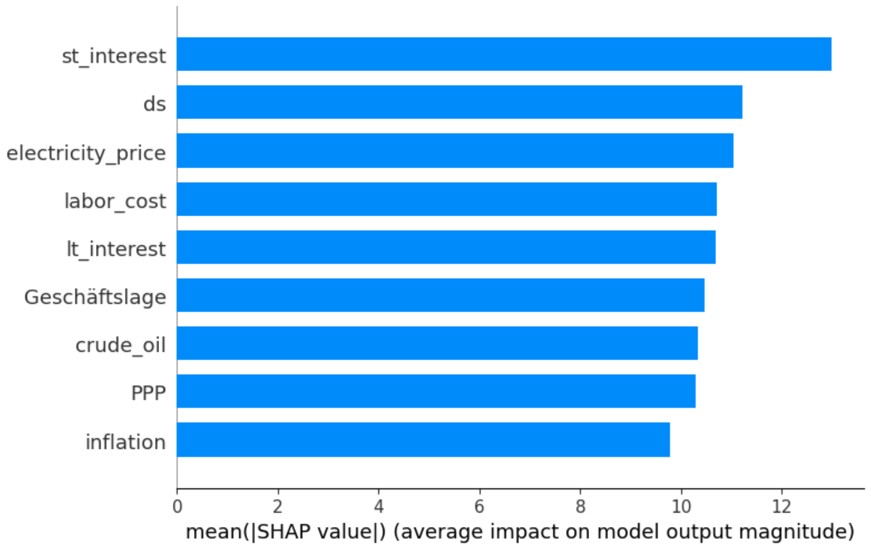


**Figure 13: SHAP plot of the major input vectors**

From this plot, we discovered that short-term interest rates have a significant impact on the final price predictions. However, despite adjusting based on this insight, the Prophet model continued to struggle with accurately capturing the aluminium price trend. This either suggests that there are additional complexities or factors specific to aluminium that the model was unable to account for effectively, or that the model simply cannot capture complex patterns.

### 3.5 Methodology Approach

NeuralProphet is an open-source forecasting tool that enhances time series predictions by integrating neural networks with Facebook's Prophet model. It is designed to handle complex patterns like seasonality, trends, and special events, offering flexibility and improved accuracy. NeuralProphet combines traditional statistical methods with neural network components, allowing for the modelling of non-linear relationships in data. It is robust against missing data, supports customizable trends and seasonality, and can incorporate auto-regressive terms to account for temporal dependencies. Scalable and Python-based, NeuralProphet is accessible for various real-world applications.

We used NeuralProphet, an evolution of the Prophet algorithm created by Facebook, which is time series prediction algorithm, we opted for Neural prophet not the original Prophet model, as it showed significant gain in efficiency because it can capture more complex patterns than the vanilla prophet model.

First, we trained separate NeuralProphet models for each variable we are interested in knowing its future value, as those aspects contribute to the final product price estimation. The variables are divided as follows, price trends of different material alloys: High carbon, medium carbon, low carbon (ST37), nodular cast iron, grey cast iron, non-alloy cast and aluminium. On top of those, we trained another Neural Prophet model to forecast labour cost.

Here are the forecasted values for aluminium over the historical price trend. Notably, the Neural Prophet model demonstrates a significantly better performance compared to the original Prophet model when it comes to predicting aluminium prices. This improvement highlights the Neural Prophet model's enhanced ability to capture the nuances and complexities of aluminium price movements.

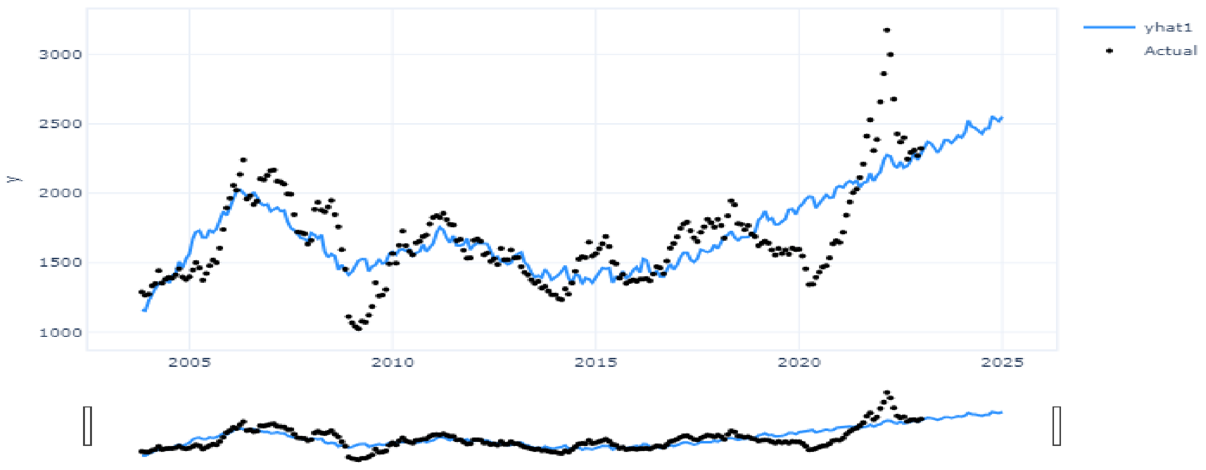


**Figure 14: Plots of the fitted NeuralProphet Model on aluminium**

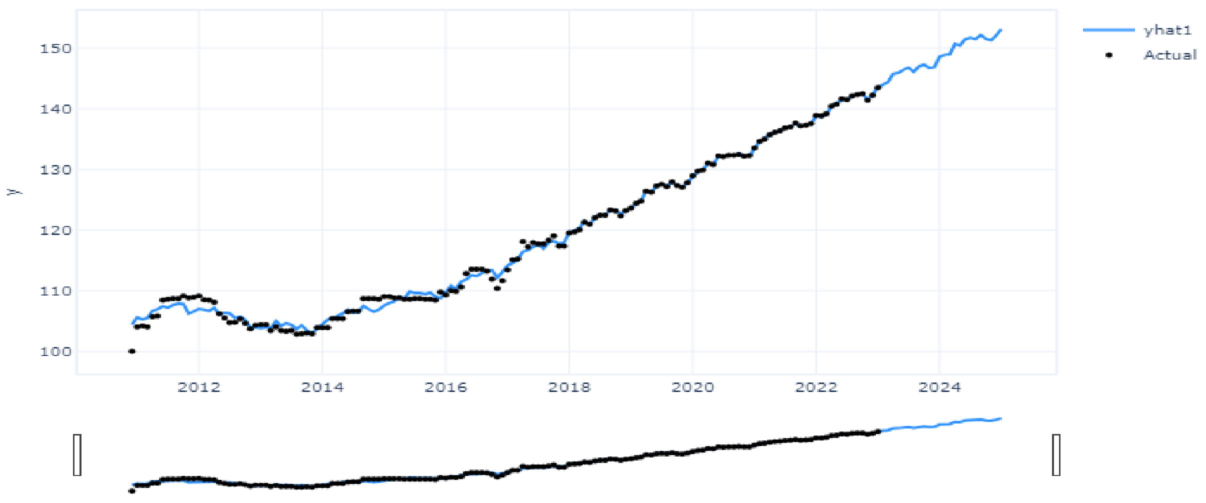


Figure 15: Plots of the fitted NeuralProphet Model on Medium Carbon

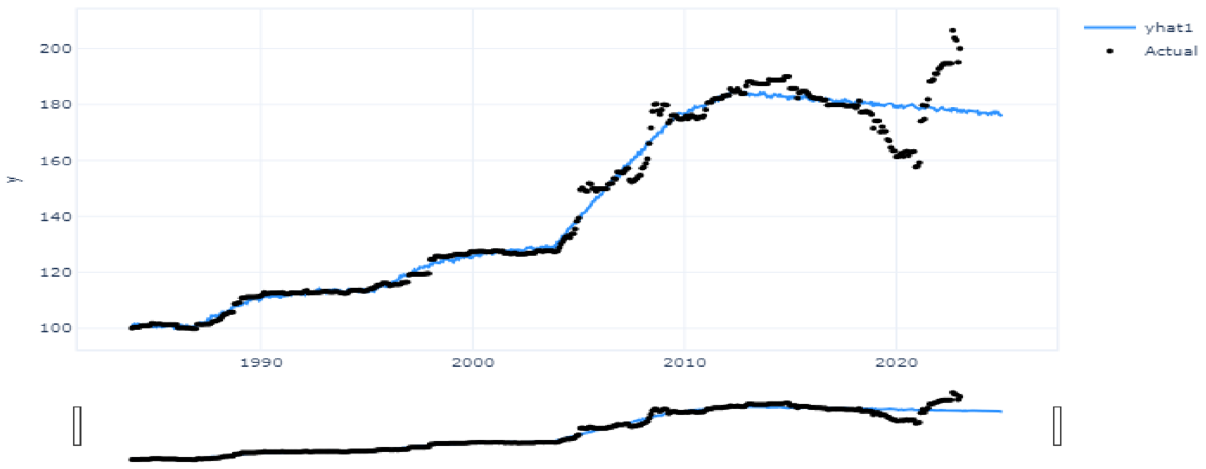


Figure 16: Plots of the fitted NeuralProphet Model on Low carbon (ST37)

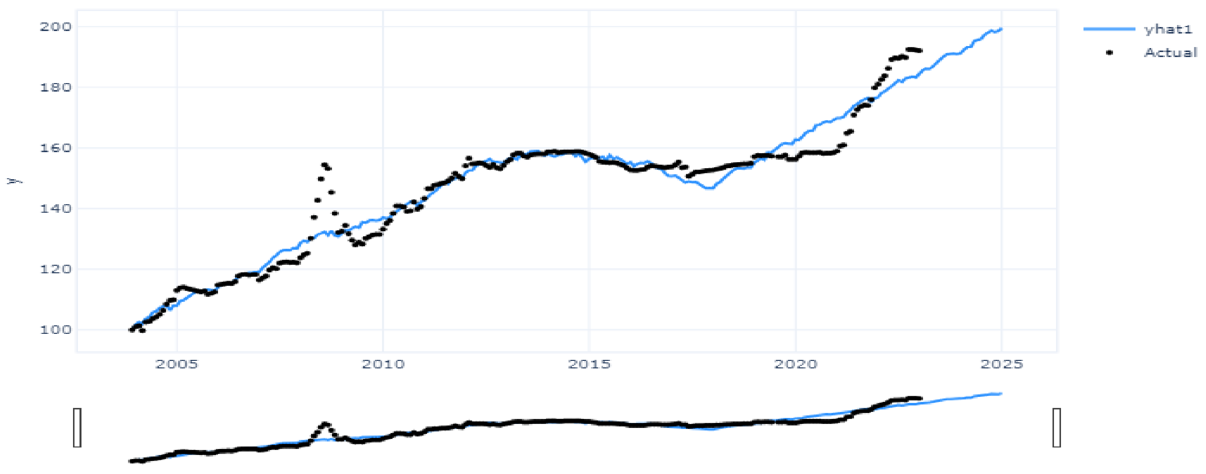


Figure 17: Plots of the fitted NeuralProphet Model on Nodular Cast Iron

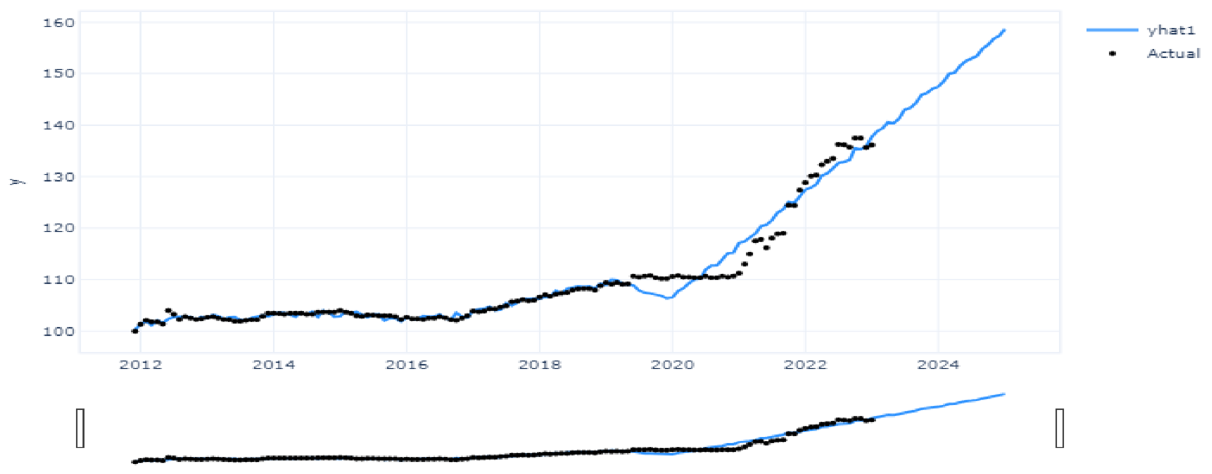


**Figure 18: Plots of the fitted NeuralProphet Model on Grey Cast Iron**

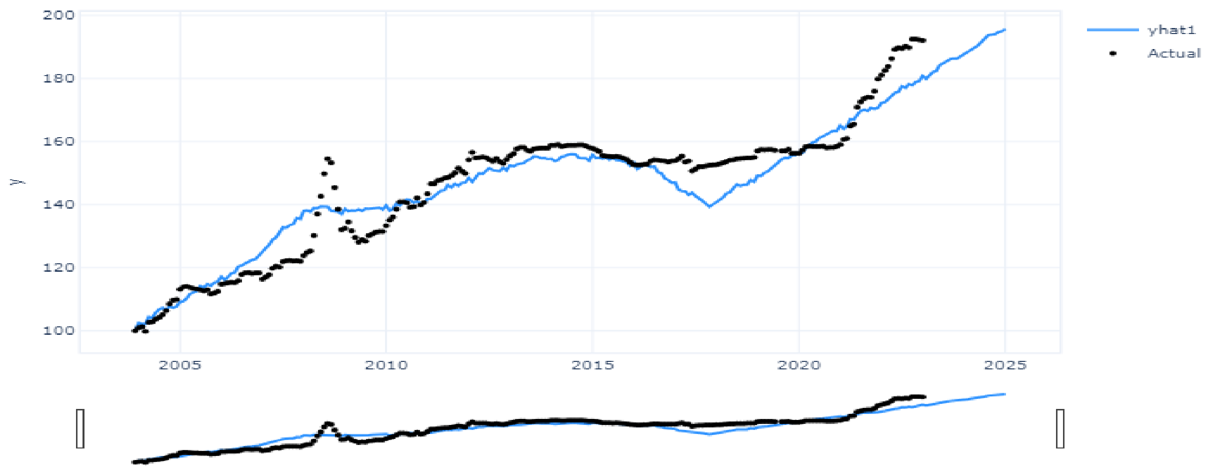


**Figure 19: Plots of the fitted NeuralProphet Model on Non-alloy Cast Iron**

We calculate the final product price based on the materials weights provided by the user to the query of the API server, but generally the final price is the accumulated multiplication of material weight with the predicted price for that year in addition to other contributing variables like labour cost. Generally, we could formulate it as follows:

$$\mathbf{P = \sum (material\ weight * forecasted\ price\ of\ material) + additional\ factors}$$

The previous equation defines a general way used to calculate prices. However, input prices used for the model training are based on index prices which allows model training while preserving company secrets such as supplier prices, market prices and alike. Using a history price trend to train the models on were referenced values (and not real priced values) preserves these company secrets as they are not leaving the company. However, the above formula may adjust for real-prices by giving the end-user the possibility to define a spot-price for each material and hereby adjusted the formula to account for the referenced price and convert them to real priced values by using the spot price of the material provided by the end user. This selected approach allows for a maximal degree of privacy as real costs are never stored and only calculated by the service on the fly using the provided service interface.

We convert the referenced price based on the year of the spot price given by the user and that becomes the new reference year and rebasing the reference value to that specific year and converting the predicted referenced values.

**Adjusted predicted values = (projected values / spot year price) * spot price**

For every predicted value we get the relative percentage change to the spot year and then multiply it by the given spot price yielding the adjusted values. However, if no spot price provided of any material the user will get a response of predicted referenced values. But the service doesn't allow the mix of spot priced with referenced values as accumulating them won't produce a valid outcome.

Then the predicted price formula becomes:

$$P = \sum (\text{material weight} * \text{Adjusted predicted values}) + \text{additional factors}$$

## 3.6 Results

The final service has been implemented as a REST service which offers the trained AI models to the InnoSale platform. A REST service is a web service that adheres to REST architecture principles, enabling communication between systems over HTTP. It is designed to be simple, stateless, and scalable, making it ideal for building APIs. REST services are structured around resources, which are entities or objects that can be accessed and manipulated via specific URLs. These services utilize standard HTTP methods to perform actions like retrieving, creating, updating, or deleting resources. Each client request is self-contained, providing all necessary information, which allows the service to be stateless and scalable.

### 3.6.1 Service Parameters

We implemented multiple functions that contributes to the overall functionality of the service that can be invoked by the end user.

#### 3.6.1.1 Status

By invoking this function call: http://innosale.sagresearch.de:8012/status, you can easily check the status of the server. This call provides real-time information, indicating whether the server is currently running or out of service.

#### 3.6.1.2 Help

When you call http://innosale.sagresearch.de:8012/help, the end user is provided with a detailed list of all the arguments that can be passed to the prediction function, along with concise explanations for each. Here's what the function call returns:

```
{"st37":"Weight of ST37 in KG","p_st37":"Spot price of
ST37","p_high_carbon":"Spot price of High Carbon","alu":"Weight of Alu in
KG","labour":"Labor hours","high_carbon":"Weight of High Carbon in
KG","medium_carbon":"Weight of Medium Carbon in
KG","p_medium_carbon":"Spot Price of Medium
Carbon","p_nodular_cast_iron":"Spot Price of Nodular Cast
Iron","nodular_cast_iron":"Weight of Nodular Cast Iron in
KG","grey_cast_iron":"Weight of Grey Cast Iron in
KG","p_grey_cast_iron":"Spot Price of Grey Cast
Iron","nonalloy_cast":"Weight of Nonalloy Cast in
KG","p_nonalloy_cast":"Spot Price of Nonalloy Cast","months":"Forecasting
period in months (default is 24 months if argument not provided)"}
```

This comprehensive return allows users to understand the parameters they can manipulate and how each one influences the prediction model.

#### 3.6.1.3 Calculate

Calculate is the core functionality of the service. For example, when calling http://innosale.sagresearch.de:8012/calculate/?copper=15&alu=12, you can input material weights, such as copper and aluminium, into the query. The function then predicts and returns the accumulated price of the product over the next two years. This powerful tool allows users to forecast costs with precision based on the materials used.

##### 3.6.1.3.1 Spot price using user-specific costs

The spot price is provided by using the spot-price flag, which is a floating-point number that precedes the alloy type. This may be the point where actual price- data is added to the request. Using this flag allows to accurately represent the current market price for a specific alloy. By prefixing the alloy with its corresponding spot-price value, the function ensures that the most up-to-date pricing is factored into the overall cost calculations. This parameter is essential for generating precise forecasts and making informed decisions based on real-time market data.

http://innosale.sagresearch.de:8012/calculate/?st37=12&months=12&p_st37=20

##### 3.6.1.3.2 Graphical Representation of price

All the above parameters can be used to plot figures and hence create a graphical representation of the price forecast. This is far easier to comprehend and improves readability of especially highly volatile prices. The graphical representation is obtained by using the “plot” command from the service.

http://innosale.sagresearch.de:8012/**plot**/?st37=500

### 3.6.2 Service Examples

The service generates a final predicted accumulated price for a product based on the supplied weights and available spot prices of the base materials used in its manufacturing. If the user provides invalid input, the service identifies the specific parameter causing the issue and returns a corresponding error message. The output is returned in JSON format.

The user provides the parameters needed to compute the final price, so for example given the weight of ST37 and want to predict for 12 months (1 year) starting from 2/2023, this could be changed after retraining the models depending on the date range of the training set. This is an example of a query that you could execute:

**http://innosale.sagresearch.de:8012/calculate/?st37=15&months=12**

Then the API will return the calculated final (referenced) price predictions for 12 months:

```
[
{"ds":"2023-02","total_product_value":2125.2080078125},
{"ds":"2023-03","total_product_value":2134.99658203125},
{"ds":"2023-04","total_product_value":2128.94873046875},
{"ds":"2023-05","total_product_value":2125.4296875},
{"ds":"2023-06","total_product_value":2133.68383789063},
{"ds":"2023-07","total_product_value":2132.35327148438},
{"ds":"2023-08","total_product_value":2134.72290039063},
{"ds":"2023-09","total_product_value":2133.21020507813},
{"ds":"2023-10","total_product_value":2129.39599609375},
{"ds":"2023-11","total_product_value":2131.46118164063},
{"ds":"2023-12","total_product_value":2114.35986328125},
{"ds":"2024-01","total_product_value":2123.82983398438}
]
```

Example for invalid query:

**http://innosale.sagresearch.de:8012/calculate/?st37=12&months=12&p_st37=20&high_carbon=13**

This previous example, the user provides a spot price for ST37 but only the weight for high carbon this won't work, if the user provides one material's spot price the user must provide spot price for all the other materials in the query.

This is the server response to the previous query:

```
{"detail":"Spot price for high_carbon is missing while other materials
have spot prices."}
```

Also, it is invalid to provide a spot price of a material without providing the material's weight:

**http://innosale.sagresearch.de:8012/calculate/?st37=12&months=12&p_st37=20&p_high_carbon=22**

Server's response:

```
{"detail":"Weight for high_carbon is missing while its spot price is
provided."}
```

Using the ***plot*** functionality, graphs that reflect the actual numbers can be sketched. Remember, that all from the afore mentioned parameters can be used for the ***plot*** command, in the same sense they are used for the ***calculate*** command.

Using the following service call, the subsequent graphic is generated:

http://innosale.sagresearch.de:8012/plot/?st37=500&copper=1&p_copper=1&labour=12

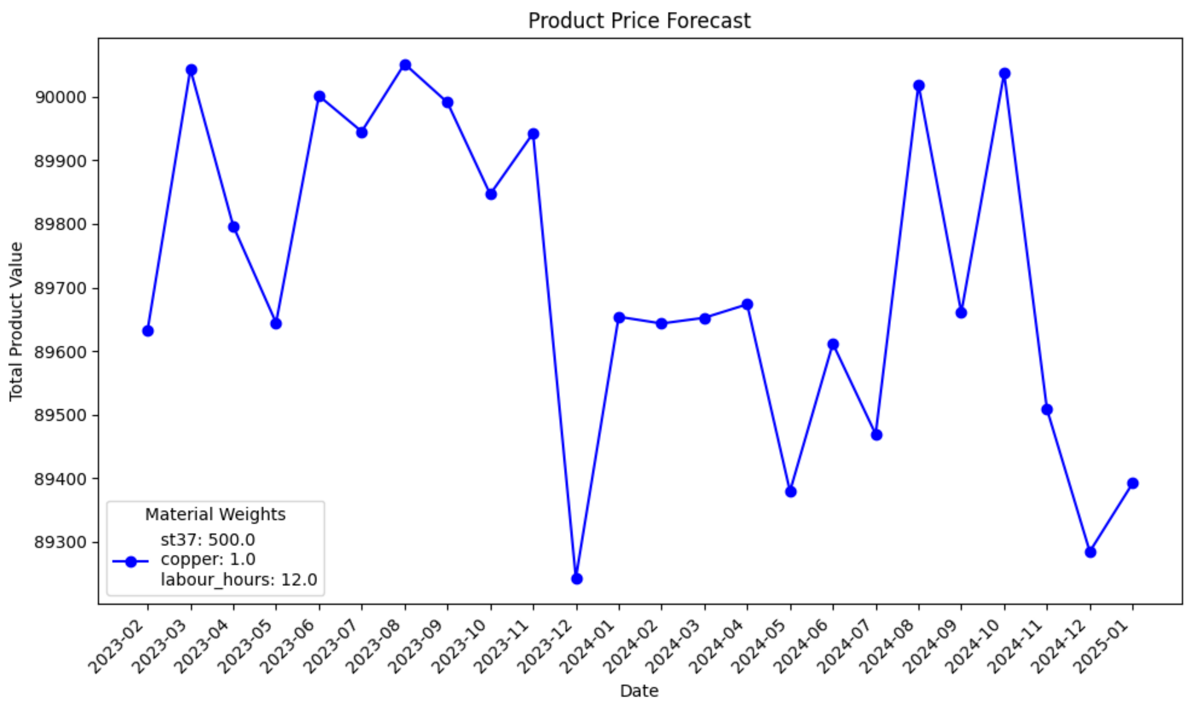


**Figure 1: Service output when using st37=500, copper=1, p_copper=1 and labour=12**

## 3.7 Outlook and Selected Approach

The implemented approach allowed us to provide a solution for the end-users without making explicit use of company-sensitive and privacy-related pricing information. Using index-prices we ensure data privacy, especially when dealing with sensitive company information like costs and pricing, and hereby circumvent the misuse and leakage of sensitive company information. Index prices represent general market trends rather than the specific financial data of any one company. When an AI model is trained using these index prices, it learns to understand broad market behaviour without ever being exposed to a particular company's sensitive data. This means that when a company inputs its own cost data into the model to generate a pricing forecast, the model uses its knowledge of the overall market to make predictions. Importantly, the model does not directly handle or retain any of the company's proprietary information, keeping that data secure.

The model is only trained on general market data, not on any specific company's costs or pricing strategies. This separation ensures that the model does not learn or store any details that could compromise a company's privacy. When a company uses the model to forecast prices, it inputs its data, but this data is used only to generate the forecast and isn't retained by the model afterward. This one-way application of data further protects sensitive information. Since the model is not built on any single company's data, the risk of leaking or exposing sensitive information is minimized. Even if someone were to access the model, they would only see generalized market trends, not specific company data. This approach greatly reduces the risk of data breaches, as the most sensitive information remains private and secure.

The model makes predictions in real-time based on the data a company provides. After the forecast is made, there is no lingering connection between the model and the company's sensitive data, which further ensures privacy. Additionally, because the model relies on index prices, it can be shared and used by multiple companies without risking the exposure of any one company’s proprietary information. Each company can safely use the model to forecast prices, knowing that their data remains private and secure.

# 4 Fuzzy Logic-based Pricing Strategies

## 4.1 Motivation

The previous chapter has already explained the extent to which adjustments to master price lists can be proposed using AI-based mechanisms. This advanced approach enables dynamic pricing, which, however, has yet to be transferred to the bill of material (BOM) level. In this context, the application of fuzzy logic is considered a crucial approach to adapt the prices of the BOM to specific factors. These factors range from the capacity utilization of the production facility to customer loyalty. This research proposes to provide the sales engineer with a toolkit based on fuzzy logic to recommend the extent to which the price of the BOM can be adjusted on a percentage basis.

Previous decisions regarding these adjustments were largely based on the individual experience of the sales engineer. However, this approach creates the possibility for sales engineers to define variables and rules that influence the final price depending on various factors. This innovative approach makes it possible to adapt discounts to customers according to the situation by enabling precise and dynamic control of pricing strategies.

## 4.2 Principles

We assume, that the price *p* of a product is to be calculated from the following formula:

$$p = f * p_s + k$$

with

$p$: final price,

$p_s$: the standard price, defined once per year,

$f$: final price factor,

$k$: a price offset.

The standard price is a given quantity and can be determined from the manufacturer's ERP system. However, the price factor and the price offset depend on a variety price influencing parameters:

$$f = \mathrm{f}\,(f_1, f_2, \ldots, f_i)$$

$$k = \mathrm{k}\,(f_1, f_2, \ldots, f_i)$$

with

$f_1, f_2, \ldots, f_i$: price influencing parameters.

In our case, the final price factor *f* and the price offset *k* are interpreted as a set of fuzzy logic rules, which is applied on the set of price influencing parameters.

**Example**:

Let's assume we have a fuzzy logic rules base like the following:

IF factory_load(high) THEN final_price_factor(quite_high).

IF customer_buys(frequently) THEN final_price_factor(quite_low).

Here, factory load and customer buys are price influencing parameters. We have to gather them from the IT systems of the manufacturer. Both rules have influence on the target variable final_price_factor. Therefore, the inference engine needs to be capable to handle conflicting formulas. It is likely, that factory_load and customer_buys but also the target variable final_price_factor have numeric value representations. The parameters need to be transformed into a representation, which is applicable for a fuzzy logic inference engine by fuzzification. Vice versa, the fuzzy logic representation of final_price_factor has to be converted back by de-fuzzification

Short literature overview

- [Bronstein, 2001]: is a pocketbook for engineers with a comprehensive chapter about fuzzy logic. In addition to a basic introduction to the conceptual world, two major types of fuzzy logic inference are described: method of Mamdani and the method of Sugeno.
- [IEC 61131-7]: IEC 61131 is the standard of programming languages for industrial programmable logic controllers (PLCs). The standard describes all necessary elements of a fuzzy logic system like fuzzy term definitions, fuzzification and defuzzification algorithms.
- [MathWorks FLT]: The Fuzzy Logic Toolbox is a commercial tool of the MathWorks corporation. The toolbox includes possibilities for modeling linguistic variables, membership functions, fuzzification and defuzzification algorithms. A graphical fuzzy logic designer is also available by MathWorks. This tool allows individuals to design, test, and tune a fuzzy inference system (FIS) for modelling complex system behaviour.

Discussion and decisions

A first decision concerns the type of inference method. According to [Bronstein, 2001] the method of Mamdani leads in the result via an independent linguistic variable to a numerical value. Sugeno's method leads directly to a numerical value via a function of all influencing input variables. Thus, Sugeno's method requires a deeper understanding of the mathematical relationships between input and output variables. Mamdani's method requires an additional inference step (defuzzification), but eliminates the need for mathematical analysis of input-output functions. According to a geeksforgeeks.org article[4] , Sugeno inference can infer a single output variable from multiple input variables, while Mamdani's method can also infer multiple output variables. Since price is composed of factors (e.g., energy price or wage increases) and offsets (e.g., transportation costs), so multiple output variables are obviously needed, Mamdani's method is preferable.

Both, the standard [IEC 61131-7] and the commercial solution [MathWorks FLT] provide broader functionality than used in the literature for pricing, which we found in our research. This makes it very flexible and candidate solutions for the pricing technologies to be used in InnoSale. However, the commercial solution is very expensive. The development of an open sourced fuzzy logic system with the functionality of e.g. [IEC 61131-7] would be an easy to use contribution to the European research community. The [IEC 61131-7] supports inference according to the Mamdani approach and thus meets the previously discussed requirement.

Figure 20 provides an overview about the Fuzzy Logic system for optimized pricing.

[4] https://www.geeksforgeeks.org/comparison-between-mamdani-and-sugeno-fuzzy-inference-system/

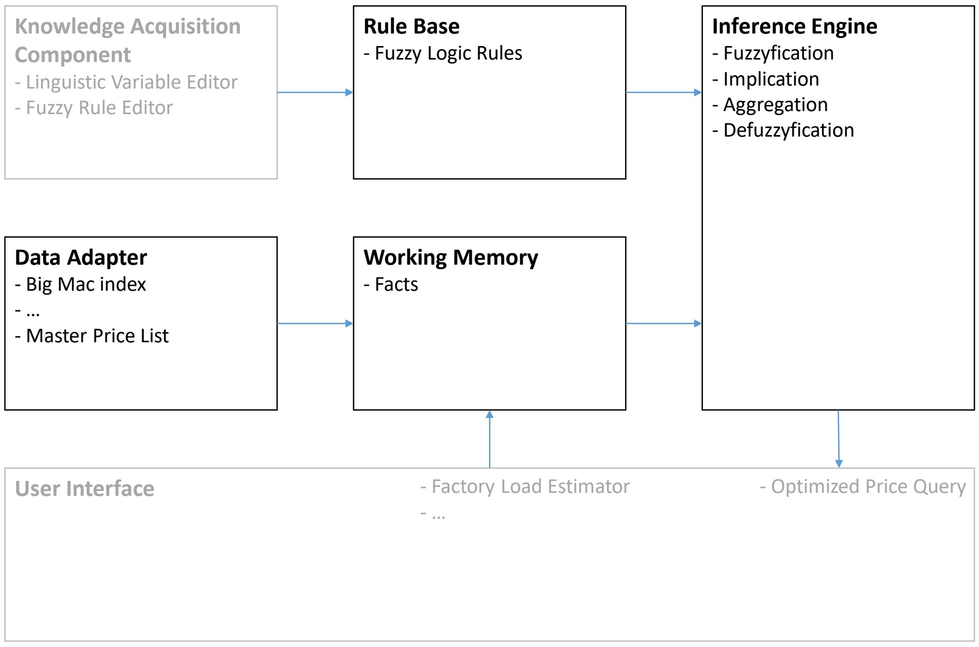


**Figure 20: Components of the Fuzzy Logic System for Optimized Pricing**

The system overview introduces following components:

- *Knowledge Acquisition Component*: this is a user frontend for the expert, who knows and defines e.g. how much discount a customer will get according to the sales history and the predicted sales with a customer.
    - Linguistic variable editor: in an algebraic system, variables are defined e.g. as integer or floating point values. In a fuzzy logic system, variables are defined as sets of linguistic terms, which themselves are defined by membership functions. Thus, this editor is a user interface for defining those things.
    - Fuzzy rule editor: this sub-component enables input of IF-THEN rules, which refer to linguistic variables and their terms.
- *Rule Base*: this component is an online store for definitions of linguistic variables and fuzzy logic rules.
- *Data Adapter*: these are separate sub-components, which acquire data as common algebraic variable values. For example, they will provide the price of a material or the Big Mac Index (BMI) as floating point values or similar.
- *Working Memory*: this component is a kind of temporal database, which contains all gathered data from the context of the price estimation.
- *User interface*: this is a frontend for the user, who want to get the price estimation. We announced two sub-components here:
    - Factory Load Estimator: in an early evaluation phase of the InnoSale project, it was decided to not use exact factory loads based on ERP entries about sales processes. Instead the factory load should be estimated e.g. weekly. Since the input is a function over time, we need a small user interface for input of it – the Factory Load Estimator.
    - Optimized Price Query: this is the interface for the sales engineer who can query prices for a product with a given bill of material (BOM).

## 4.3 Results

### 4.3.1 Introduction

A Fuzzy Control Language Engine (FCLE) has been developed, which is an innovative tool for Fuzzy Logic applications, primarily designed for dynamic pricing systems. Adhering to the IEC 61131-7 standard, FCLE offers versatile use across various industries. It supports flexible rule management and seamless integration, enabling users to customize control strategies by easily modifying rule bases and input configurations. Beyond dynamic pricing, FCLE is suitable for diverse applications requiring fuzzy logic-based control algorithms.

Here is a more detailed list of the features of the FCLE:

- **IEC 61131-7 Compliance**: Is compatible to a subset of the IEC 61131-7 standard for Fuzzy Control Language, allowing for standardized evaluation of function blocks.
- **Comprehensive RESTful API**: Offers a robust API for configuring rule bases, input values, and dynamically triggering the evaluation of fuzzy logic rules.
- **Easy Installation and Setup**: Designed for straightforward installation with minimal configuration, enabling quick deployment.
- **Integration Ready**: Seamlessly integrates with existing GUIs and external systems, suitable for a wide range of industrial applications beyond dynamic pricing.
- **Flexible Rule Base Management**: Supports uploading, modifying, and retrieving rule bases in a standardized format, enhancing adaptability for various control strategies.
- **Dynamic Input Handling**: Capable of processing and evaluating input values from multiple sources, ensuring robust and responsive control algorithms.

An example Fuzzy Logic file, which is used in the test cases of the FCLE, looks like this:

```
FUNCTION_BLOCK DynamicPricing

    VAR_INPUT
        factory_load: REAL;   // Represents the current load on the factory (0 to 100%)
        market_demand: REAL;  // Represents the market demand level (0 to 100%)
        production_cost: REAL; // Represents the cost of production (0 to 100%)
    END_VAR

    VAR_OUTPUT
        pricing_factor: REAL; // The output factor used for dynamic pricing
    END_VAR

    FUZZIFY factory_load
        TERM low := (0, 1) (30, 1) (60, 0);
        TERM medium := (30, 0) (50, 1) (70, 0);
        TERM high := (60, 0) (100, 1);
    END_FUZZIFY

    FUZZIFY market_demand
        TERM low := (0, 1) (25, 1) (50, 0);
        TERM medium := (25, 0) (50, 1) (75, 0);
        TERM high := (50, 0) (100, 1);
    END_FUZZIFY

    FUZZIFY production_cost
        TERM low := (0, 1) (20, 1) (40, 0);
        TERM medium := (20, 0) (50, 1) (80, 0);
        TERM high := (60, 0) (100, 1);
    END_FUZZIFY

    DEFUZZIFY pricing_factor
        TERM low := 0;
        TERM medium := 50;
        TERM high := 100;
        METHOD: CoGS;
        DEFAULT := 0;
    END_DEFUZZIFY

    RULEBLOCK No1
        AND: MIN;
        ACCU: MAX;

        RULE 1: IF factory_load IS high THEN pricing_factor IS high;
        RULE 2: IF market_demand IS high THEN pricing_factor IS high;
        RULE 3: IF production_cost IS high THEN pricing_factor IS high;
        RULE 4: IF factory_load IS medium AND market_demand IS medium
                THEN pricing_factor IS medium;
        RULE 5: IF factory_load IS low AND market_demand IS low AND production_cost IS low
                THEN pricing_factor IS low;
        RULE 6: IF factory_load IS medium AND market_demand IS low
                THEN pricing_factor IS medium;
        RULE 7: IF production_cost IS medium THEN pricing_factor IS medium;

    END_RULEBLOCK

END_FUNCTION_BLOCK
```

The FCLE is developed in Python and consists basically of 4 main modules:

- **engine.py**: Serves as the core component of the FCLE by managing the definition and manipulation of fuzzy logic elements. It encapsulates the creation of input and output variables, handles the fuzzification and defuzzification processes, and orchestrates the evaluation of fuzzy rules through comprehensive classes such as InputVariable, OutputVariable, Rule, and FuzzyFunctionBlock. The Engine class integrates these elements to facilitate the overall fuzzy logic reasoning required for dynamic pricing and other control applications.
- **parser.py**: Implements the parsing mechanism for Fuzzy Control Language (FCL) files adhering to the IEC61131-7 standard. Utilizing the Lark parsing library, it defines a specific grammar to accurately interpret FCL syntax and transform it into executable fuzzy function blocks. The FCLTransformer class systematically converts parsed FCL structures into the internal representations used by the engine, ensuring seamless integration of rule bases and variable configurations into the fuzzy logic framework.
- **service.py**: Establishes a FastAPI-based service interface that exposes the functionality of the FCLE to external clients. It provides RESTful endpoints for evaluating fuzzy logic rules (/evaluate), managing rule bases (/set_rulebase and /get_rulebase), and configuring service settings (/set_service_conf and /get_service_conf). The module initializes the fuzzy logic engine, handles HTTP requests, processes input parameters, and returns structured responses, thereby enabling real-time interaction and integration with other systems or user interfaces.
- **io_connectors.py**: Designed to facilitate the acquisition of missing variable values through API calls, this module is intended to enhance the engine's ability to dynamically retrieve necessary data inputs. Currently, this functionality remains unimplemented and is slated for development during complex testing phases at demonstrator sites. Once integrated, io_connectors.py will enable the FCLE to autonomously fetch and update variable values from external sources, thereby increasing its robustness and adaptability in diverse operational environments.

### 4.3.2 Quick Start

**1. Clone the Repository**

Currently, the software is hosted at ifak. Thus proceed as follows:

```
git clone https://code.ifak.eu/mth/innosale_fuzzylogic_engine.git
cd innosale_fuzzylogic_engine
```

**2. Set Up the Environment**

*Automated Installation*

For Linux-based systems (including WSL on Windows), run:

```
bash ./bin/install.sh
```

This script will:

- Create a Python virtual environment.
- Upgrade pip to the latest version.
- Install the package in editable mode.
- Install all required dependencies.

*Manual Installation*

If you encounter issues with the automated script, you can install manually:

```
python -m venv --copies venv
source venv/bin/activate
python -m pip install --upgrade pip
python -m pip install -e .
pip install -r requirements.txt
```

**Note:** The `requirements.txt` may not always contain the latest versions of dependencies.

**3. Run the Service**

Start the FastAPI server:

```
fcle
```

The service will be accessible at `http://0.0.0.0:8000`. You can interact with it via:

- Web Browser: Navigate to `http://localhost:8000/docs` for interactive API documentation.
- Command-Line Tools: Use tools like `curl` or `httpie` to make HTTP requests.
- External Devices: Access via any device on the same network using the host's IP address and port `8000`.

To stop the server, press `Ctrl+C`.

### 4.3.3 Installation

Ensure you have Python 3.10 or higher installed. The installation steps are covered in the Quick Start section above.

### 4.3.4 Configuration

Before running the service, ensure the `data` directory contains:

- `rulebase.fcl`: The initial rule base file.
- `service_conf.yaml`: Service configuration file.

You can modify these files or use the API endpoints to update them dynamically.

**Note:** Integration with Keycloak or other authentication mechanisms is not supported out of the box. Modifications to `service.py` are required for such features.

### 4.3.5 Usage

Once the service is running, you can interact with it using HTTP requests. The service provides endpoints to evaluate fuzzy logic rules, manage rule bases, and update service configurations.

### 4.3.6 API Documentation

Interactive API documentation is available at:

- Swagger UI: http://localhost:8000/docs
- Redoc: http://localhost:8000/redoc

These interfaces provide detailed information about each endpoint, expected parameters, and response formats. The OpenAPI specification is also available at http://localhost:8000/redoc.

### 4.3.7 Example API Calls

Below are detailed examples of how to interact with the FCLE API using `curl`. Replace `localhost` with your server's address if it's running elsewhere.

**1. Evaluate Endpoint**

**Endpoint:** `POST /evaluate`

Evaluates the fuzzy logic rules based on provided input values.

**Example Request:**

```
curl -X POST "http://localhost:8000/evaluate" \
     -H "Content-Type: application/json" \
     -d '{"input_values": {"factory_load": 75, "market_demand": 60,
"production_cost": 50}}'
```

**Note**: Make sure that those 3 input values are defined in the rulebase. If necessary parameters are missing, you will get an error.

**Headers:**

`Content-Type: application/json`

**Body:**

`input_values`: JSON object containing input variables.

**Example Response:**

```
{
  "status": "success",
  "timestamp": "2024-09-02T12:34:56Z",
  "request_id": "abcd1234",
  "input_params": {
    "factory_load": 75,
    "market_demand": 60,
    "production_cost": 50
  },
  "output_values": {
    "pricing_factor": 1.23
  },
  "message": "Pricing factor successfully calculated."
}
```

**Explanation:**

- Input Variables:
  - `factory_load`: Current load on the factory (0 to 100%, or more with extra shifts).
  - `market_demand`: Market demand level (0 to 100, 50 is normal demand).
  - `production_cost`: Cost of production (0 to 100, 50 is normal production cost).
- Output Variables:
  - `pricing_factor`: Calculated pricing factor based on fuzzy logic rules.

## 2. Set Rulebase Endpoint

**Endpoint:** `POST /set_rulebase`

Uploads a new rule base in FCL (Fuzzy Control Language) format.

**Example Request:**

```
curl -X POST "http://localhost:8000/set_rulebase" \
     -H "Content-Type: application/json" \
     -d '{"rulebase": "'$(base64 -w 0 path_to_your_rulebase.fcl)'"}'
```

**Headers:**

`Content-Type: application/json`

**Body:**

`rulebase`: Base64-encoded string of your `rulebase.fcl` file.

**Example Response:**

```
{
  "status": "success",
  "timestamp": "2024-09-02T12:35:10Z",
  "message": "Rule base successfully updated."
}
```

**Explanation:**

This endpoint replaces the existing rule base with the provided one.

Ensure your FCL file is correctly formatted and encoded.

## 3. Get Rulebase Endpoint

**Endpoint:** `GET /get_rulebase`

Retrieves the current rule base in use.

**Example Request:**

```
curl -X GET "http://localhost:8000/get_rulebase"
```

**Example Response:**

```
{
  "status": "success",
  "timestamp": "2024-09-02T12:35:30Z",
  "rulebase": "<base64_encoded_rulebase>",
  "message": "Rule base successfully retrieved."
}
```

**Explanation:**

The `rulebase` field contains the Base64-encoded FCL content.

Decode using:

```
echo "<base64_encoded_rulebase>" | base64 -d > retrieved_rulebase.fcl
```

**4. Set Service Configuration Endpoint**

**Endpoint:** `POST /set_service_conf`

Updates the service configuration.

**Example Request:**

```
curl -X POST "http://localhost:8000/set_service_conf" \
     -H "Content-Type: application/json" \
     -d '{"service_conf": "'$(base64 -w 0
path_to_your_service_conf.yaml)'"}'
```

**Headers:**

`Content-Type: application/json`

**Body:**

`service_conf`: Base64-encoded string of your `service_conf.yaml` file.

**Example Response:**

```
{
  "status": "success",
  "timestamp": "2024-09-02T12:35:50Z",
  "message": "Service configuration successfully updated."
}
```

**Explanation:**

- Updates the service configuration parameters.
- Useful for changing server settings or variable endpoints.

**5. Get Service Configuration Endpoint**

**Endpoint:** `GET /get_service_conf`

Retrieves the current service configuration.

**Example Request:**

```
curl -X GET "http://localhost:8000/get_service_conf"
```

**Example Response:**

```
{
  "status": "success",
  "timestamp": "2024-09-02T12:36:10Z",
  "service_conf": "<base64_encoded_service_conf>",
  "message": "Service configuration successfully retrieved."
}
```

**Explanation:**

The `service_conf` field contains the Base64-encoded YAML content.

Decode using:

```
echo "<base64_encoded_service_conf>" | base64 -d >
retrieved_service_conf.yaml
```

### 4.3.8 Testing

To run the test suite:

```
bash ./bin/test.sh
```

Or manually:

```
source venv/bin/activate
python -m unittest discover test
```

**Note**: You can also take the test cases as examples and for getting more insights on how to use the API.

# 5 Conclusion

The research presented in **section 2** of this report highlights the development of innovative 3D Shape-based Pricing Strategies designed to enhance cost estimation accuracy in industries such as metal sheet stamping and mold-making. These strategies leverage advanced machine learning models and geometric analyses, resulting in significant improvements over traditional methods.

Key achievements include:

- Enhanced Cost Prediction Accuracy: By integrating voxelization techniques and other geometric analyses, the 3D shape-based approach reduces mean absolute percentage error (MAPE) from traditional methods to around 10% using LightGBM and XGBoost models. This improvement is particularly evident in complex custom part orders.
- Reduced Quotation Time: The automation of cost estimation through machine learning models has significantly reduced the time required for generating quotations. This reduction in quotation time is crucial in industries where speed is a competitive advantage.
- Transparency and Customer Satisfaction: By providing more transparent pricing models based on data-driven predictions, companies can offer clearer justifications for their pricing decisions. This increased transparency has been shown to improve customer satisfaction and trust.

The research presented in **section 3** of this report highlights the development of a novel AI-based system for pricing strategies that leverages statistical analysis and forecasting to automate and optimize the adjustment of master price lists. The approach aims to mitigate the time-intensive, manual processes currently employed by product managers while enhancing accuracy and responsiveness to market conditions.

Key achievements include:

- Automated Forecasting: By integrating real-time data from various sources such as material prices, financial indicators, and energy costs, our system can generate accurate forecasts for material price trends. This automation allows for swift adjustments in pricing based on dynamic market conditions.
- Enhanced Model Performance: The employment of NeuralProphet over the traditional Prophet model demonstrated significant improvements, particularly in capturing complex patterns like those seen in aluminium prices. NeuralProphet’s ability to integrate neural networks with statistical methods has proven effective in handling non-linear relationships and improving forecast accuracy.
- Real-Time Pricing Transparency: The system provides real-time access to price developments, enabling customers to understand the rationale behind pricing decisions, thereby fostering trust and customer satisfaction. By offering clear arguments for any adjustments, the service promotes a more transparent and customer-centric approach.
- Data Privacy Preservation: Utilizing index prices ensures that sensitive company information remains secure while still capturing general market trends essential for accurate forecasting. This method allows companies to benefit from AI-driven insights without risking data privacy or confidentiality.

The research discussed in **section 4** of this report demonstrates the potential of fuzzy logic-based pricing strategies in enhancing dynamic pricing mechanisms, particularly at the Bill of Material (BOM) level. The development and implementation of a Fuzzy Control Language

Engine (FCLE), which aligns with IEC 61131-7 standards, offer a robust and flexible solution for automating price adjustments based on various influencing factors such as factory load, market demand, and production costs.

The FCLE provides a comprehensive RESTful API that enables seamless integration into existing systems. Its modular design, encompassing the engine.py, parser.py, service.py, and io_connectors.py modules, facilitates ease of use and scalability. The system's capability to adapt pricing strategies dynamically through fuzzy logic rules reduces reliance on manual adjustments by sales engineers, thereby improving efficiency and consistency in pricing.

The research confirms that Mamdani's inference method is preferable for this application due to its ability to handle multiple output variables, which are necessary for calculating both the price factor and offset.

Future work will focus on integrating external data acquisition capabilities through io_connectors.py, enhancing the FCLE's adaptability in diverse operational environments. Additionally, extending testing phases at demonstrator sites is anticipated to further refine and validate the system's performance under real-world conditions.

# 6 Abbreviations

| **Abbreviation** | **Description** |
|---|---|
| ABAC | Attribute-Based Access Control |
| API | Application Programming Interface |
| CMMS | Computerized Maintenance Management System |
| CRM | Customer Relationship Management |
| CSV | Comma-Separated Values |
| DOM | Document Object Model |
| EDI | Electronic Data Interchange |
| ERP | Enterprise Resource Planning |
| ESB | Enterprise Service Bus |
| FCLE | Fuzzy Control Language Engine |
| GUI | Graphical User Interface |
| HTTPS | Hypertext Transfer Protocol Secure |
| HTTP | Hypertext Transfer Protocol |
| IEC | International Electrotechnical Commission |
| JSON | JavaScript Object Notation |
| JWT | JSON Web Token |
| MAE | Mean Absolute Error |
| MAPE | Mean Absolute Percentage Error |
| MOM | Message-Oriented Middleware |
| RBAC | Role-Based Access Control |
| REST | Representational State Transfer |
| SAML | Security Assertion Markup Language |
| SMOTE | Synthetic Minority Oversampling Technique |
| SSO | Single Sign-On |
| SSL | Secure Sockets Layer |
| SOA | Service-Oriented Architecture |
| TLS | Transport Layer Security |
| XML | eXtensible Markup Language |
| YAML | YAML Ain't Markup Language |

# 7 References

| | |
|---|---|
| [Bazaz2016] | Tayibia Bazaz; Aqeel Khalique: A Review on Single Sign on Enabling Technologies and Protocols. International Journal of Computer Applications, vol. 151 – No. 11, October 2016. Retrieved 2023-04-13 from https://www.researchgate.net/publication/309225903_A_Review_on_Single_Sign_on_Enabling_Technologies_and_Protocols. |
| [Erl2017] | Erl, Thomas. Service-Oriented Architecture: Analysis and Design for Services and Microservices. Prentice Hall, 2017. |
| [FastAPI] | Tiangolo (Sebastián Ramírez): FastAPI. Retrieved 2023-02-14, from https://fastapi.tiangolo.com/lo/ |
| [Fielding2000] | Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral dissertation). University of California, Irvine. Retrieved 2023-05-10 from https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. |
| [Hohpe2003] | Gregor Hohpe and Bobby Woolf. "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions". Addison-Wesley Professional, October 2003. |
| [HTTP1.1] | R. Fielding, Ed.: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, RFC 7231. Internet Engineering Task Force (IETF), June 2014. Retrieved 2023-04-28 from https://datatracker.ietf.org/doc/html/rfc7231. |
| [HTTPS] | Rescorla, E.: . HTTP Over TLS, RFC 2818. Internet Engineering Task Force (IETF), May 2000. Retrieved 2023-01-11 from https://datatracker.ietf.org/doc/html/rfc2818 |
| [JWT] | Jones, M., Bradley, J., & Sakimura, N.:. JSON Web Token (JWT), RFC 7519. Internet Engineering Task Force (IETF), May 2015. Retrieved 2022-12-12 from https://datatracker.ietf.org/doc/html/rfc7519 |
| [Keycloak] | Keycloak. Retrieved 2023-04-11 from https://www.keycloak.org/. |
| [OpenAPI] | OpenAPI Initiative. OpenAPI specification (version 3.1.0). February 2021. Retrieved 2023-04-28 from https://spec.openapis.org/oas/v3.1.0. |
| [Python] | Python Software Foundation: Python.org. Retrieved 2023-05-02, from https://www.python.org/ |
| [PyPi] | Python-Community: PyPI - the Python Package Index. Retrieved 2023-03-23, from https://pypi.org/ |
| [RBAC] | Serban I. Gavrila: Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management. National Institute of Standards and Tech., 1998. Retrieved 2023-02-16 from https://dl.acm.org/doi/10.1145/286884.286902 |
| [Sandhu1996] | R. Sandhu et al, Role-Based Access Control Models, IEEE Computer, 29(2):38-47, Feb. 1996, Retrieved 2023-02-16 from https://profsandhu.com/journals/computer/i94rbac(org).pdf |
| [Xin2012] | Xin Jin, Ram Krishnan and Ravi Sandhu: A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC. In Proceedings 26th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec 2012), Paris, France, July 11-13, 2012, pages 41-55. |

| | |
|---|---|
| [Hu2013] | Vincent C. Hu et. al.: Guide to Attribute Based Access Control (ABAC) Definition and Considerations (Draft). National Institute of Standards and Technology, April 2013. 2023-02-16 from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=cab698a5b0949aa7acd0858b55352c5df0a2c2fb |
| [SMOTE] | Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." *Journal of artificial intelligence research* 16 (2002): 321-357. |
| [Tomek] | I. Tomek, "Two modifications of CNN," In Systems, Man, and Cybernetics, IEEE Transactions on, vol. 6, pp 769-772, 1976. |
| [XGBoost] | Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016. |
| [LightGBM | Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems* 30 (2017). |
| [SHAP] | Scott, M., and Lee Su-In. "A unified approach to interpreting model predictions." *Advances in neural information processing systems* 30 (2017): 4765-4774. |