Tuomas Talonpoika

Machine Learning for Predicting Production Lead Times using ERP Data

A predictive modelling study in make-to-order manufacturing

School of Technology and Innovations
Master's thesis in Information Systems Science

Vaasa 2026

UNIVERSITY OF VAASA
School of Technology and Innovations
Author: Tuomas Talonpoika
Title of the Thesis: Machine Learning for Predicting Production Lead Times using ERP Data : A predictive modelling study in make-to-order manufacturing
Degree: Master of Science in Economics and Business Administration
Programme: Information Systems
Supervisor: Timo Mantere
Year: 2026 **Pages:** 92

ABSTRACT:
Accurate prediction of production lead times is a critical challenge for manufacturing companies, as it directly impacts delivery reliability, resource management, and customer satisfaction. Traditional forecasting methods may fail to incorporate the complex product characteristics that influence production duration. Machine learning can provide new opportunities to analyze large amounts of data and capture nonlinear relationships that can help with better prediction accuracy.

This study investigates whether machine learning models can effectively predict production lead times using historical data from ERP system, with a focus on understanding both the accuracy of such models and the factors that affect most predictions. Three explainable tree-based ensemble algorithms: Random Forest, XGBoost, and LightGBM are evaluated, and SHAP (Shapley Additive exPlanations) is used to interpret model outputs and identify which product specific features have the strongest influence on production lead time.

The research follows a Design Science Research methodology and is conducted in collaboration with an electric motor manufacturing company. Historical production data from 2019 to 2024, comprising approximately 200,000 production orders, including product specifications and realized lead times, is used to train and validate the machine learning models using time-series cross-validation and hold-out. Products are grouped into manufacturing-relevant categories, and model performance is assessed using R-squared, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). SHAP analyses help interpret the predictions of the models.

Results indicate that machine learning can provide substantially more accurate production lead time estimates than traditional planning practices. The findings reveal which algorithm performs the best and highlight the product characteristics that most significantly affect production duration. The study contributes to both practice and theory by developing a validated construction that integrates explainable machine learning into production planning and increasing understanding of interpretable AI approaches in a manufacturing context, while acknowledging that findings are specific to the case company and may not generalize to other manufacturing contexts.

KEYWORDS: Machine learning, lead time, production control, enterprise resource planning, data analytics, predictive model, design science research

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen akateeminen yksikkö
Tekijä: Tuomas Talonpoika
Tutkielman nimi: Machine Learning for Predicting Production Lead Times using ERP Data : A predictive modelling study in make-to-order manufacturing
Tutkinto: Kauppatieteiden maisteri
Oppiaine: Tietojärjestelmätiede
Työn ohjaaja: Timo Mantere
Valmistumisvuosi: 2026 **Sivumäärä:** 92

ABSTRAKTI:
Tuotannon läpimenoaikojen tarkka ennustaminen on valmistusyrityksille keskeinen haaste, sillä se vaikuttaa suoraan toimitusvarmuuteen, resurssien hallintaan ja asiakastyytyväisyyteen. Perinteiset ennustusmenetelmät eivät välttämättä kykene huomioimaan kaikkia tuotannon kestoon vaikuttavia tekijöitä. Koneoppiminen voi tarjota uusia mahdollisuuksia analysoida suuria datamääriä ja tunnistaa epälineaarisia suhteita, jotka voivat auttaa parantamaan ennusteiden tarkkuutta.

Tässä tutkimuksessa selvitetään, voivatko koneoppimismallit ennustaa tuotannon läpimenoaikoja tehokkaasti ERP-järjestelmästä saadun historiallisen datan avulla. Tutkimuksessa keskitytään ymmärtämään, sekä mallien tarkkuutta, että ennusteisiin eniten vaikuttavia tekijöitä. Tutkimuksessa arvioidaan kolmea selitettävää puupohjaista koostealgoritmia: Random Forest, XGBoost ja LightGBM. SHAP-menetelmää (Shapley Additive exPlanations) käytetään tulkitsemaan mallien tuloksia ja tunnistamaan, mitkä tuotekohtaiset ominaisuudet vaikuttavat voimakkaimmin tuotannon läpimenoaikaan.

Tutkimus noudattaa suunnittelutieteellistä tutkimusotetta ja on toteutettu yhteistyössä sähkömoottoreita valmistavan yrityksen kanssa. Koneoppimismallien opettamiseen ja validointiin käytetään historiallista tuotantodataa vuosilta 2019–2024. Aineisto koostuu noin 200 000 tuotantotilauksesta, sisältäen tuotespesifikaatiot sekä toteutuneet läpimenoajat. Validointimenetelminä hyödynnetään aikasarjojen ristiin validointia sekä hold-out menetelmää. Tuotteet on ryhmitelty valmistuksen kannalta olennaisiin kategorioihin ja mallien suorituskykyä arvioidaan selitysasteen (R^2), keskimääräisen absoluuttisen virheen (MAE) sekä neliöllisen keskihavainnoin (RMSE) avulla. SHAP-analyysit auttavat tulkitsemaan mallien antamia ennusteita.

Tulokset osoittavat, että koneoppiminen voi tarjota huomattavasti tarkempia tuotannon läpimenoaikojen arvioita kuin perinteiset suunnittelukäytännöt. Löydökset paljastavat parhaiten suoriutuvan algoritmin ja korostavat niitä tuoteominaisuuksia, jotka vaikuttavat merkittävimmin tuotannon kestoon. Tutkimus edistää sekä käytäntöä että teoriaa kehittämällä validoidun rakenteen, joka integroi selitettävän koneoppimisen osaksi tuotannonsuunnittelua, sekä lisää ymmärrystä tulkittavan tekoälyn lähestymistavoista tuotantoympäristössä, huomioiden samalla, että tulokset ovat yrityskohtaisia eivätkä välttämättä ole yleistettävissä muihin tuotantoympäristöihin.

AVAINSANAT: Koneoppiminen, läpimenoaika, tuotannonohjaus, toiminnanohjausjärjestelmä, data-analytiikka, ennustemalli, suunnittelutieteen tutkimus

Contents

Figures

Tables

Abbreviations

AI = Artificial Intelligence

DSR = Design Science Research

DSRM = Desing Science Research Methodology

ERP = Enterprise Resource Planning

GOSS = Gradient-based One-Side Sampling

IEC = International Electrotechnical Commission

LightGBM = Light Gradient Boosting Machine

LIME = Local Interpretable Model-agnostic Explanations

LT = Lead Time

MAE = Mean Absolute Error

MAPE = Mean Absolute Percentage Error

MES = Material Requirements Planning

ML = Machine Learning

MRP = Material Requirements Planning

MTO = Make-to-Order

MTS = Make-to-Stock

RF = Random Forest

RMSE = Root Mean Square Error

RQ = Research Question

SHAP = Shapley Additive exPlanations

TPE = Tree-structured Parzen Estimator

XAI = Explainable Artificial Intelligence

XGBoost = Extreme Gradient Boosting

1 Introduction

Production lead time is a key factor in manufacturing processes, as understanding its true duration is essential for effective planning. Predicting lead time accurately in manufacturing enables better production planning, optimal allocation of resources, and improved delivery reliability. Particularly in make-to-order manufacturing, where products contain various configurations, lead time management becomes more challenging. Traditional forecasting methods often rely on historical averages or simple statistical models that may not consider for more complex factors affecting the lead time. Machine learning can offer a new approach by learning complex, nonlinear relationships from large datasets without explicit programming.

This study is conducted in collaboration with a case company that is a manufacturing company that produces electric motors with various product configurations. The company uses an ERP system containing comprehensive historical production order data. For this study, data from 2019 to 2024 is utilized, comprising approximately 200,000 rows of order data, providing a substantial foundation for data-driven study.

The case company relies on traditional planning methods to estimate production lead times for different product configurations. However, as the number of different product configurations has increased, the configurations have also become more complex and diverse. The conventional approaches may no longer capture the complex relationships between the product characteristics and actual production lead time. Discussions within the case company indicated that these complex product configurations are believed to significantly affect the production lead time, though the magnitude of these effects is not well quantified in the current planning system. This uncertainty complicates production planning, resource allocation, and the ability to provide realistic delivery commitments. Advancements in machine learning and data analytics offer new opportunities to handle such complexity in production planning, yet these capabilities remain unused in the current situation.

The main challenge is predicting production lead time accurately while understanding the underlying causes. Traditional forecasting methods may struggle with the nonlinear interactions between complex product configurations. Furthermore, understanding why certain product features require longer lead times remains challenging and limits the ability to address possible production bottlenecks.

This study addresses three research questions:
RQ1: How effectively can machine learning models predict production lead times using ERP data?
RQ2: Which machine learning algorithms provide the most accurate predictions for production lead times?
RQ3: Which product configurations have the most significant impact on production lead time?

This study makes two main contributions. First, it develops and evaluates a machine learning-based artifact that can predict production lead times more accurately than the company's current planning methods, demonstrating how explainable machine learning can be integrated into production planning. Second, by applying SHAP analysis, the study identifies which product configurations most significantly affect production duration, addressing a gap in the literature where product-specific features have been largely overlooked as predictive features.

The thesis structure follows the Design Science Research process. Chapter 2 presents a literature review covering production lead time management, machine learning applications in manufacturing, and explainable AI methods. Chapter 3 details the methodology, including the research approach, case selection rationale, data collection procedures, and evaluation criteria. Chapter 4 describes data processing and feature engineering. Chapter 5 presents the artifact development, including algorithm configuration, validation strategy, and hyperparameter optimization. Chapter 6 reports the results and

evaluates the artifact against the defined criteria. Chapter 7 discusses the findings, their practical and theoretical implications, and limitations. Chapter 8 concludes the thesis.

2 Literature Review

This chapter provides the theoretical foundation for the study. Accurate lead time estimation is difficult in make-to-order manufacturing because products vary significantly and traditional planning methods rely on fixed averages that do not reflect this variation. This literature review is split into five areas: production lead time in manufacturing environments, machine learning in manufacturing, machine learning algorithms used in this study, feature engineering, and explainable AI methods.

2.1 Production Lead Time in Manufacturing Environments

Production lead time is a key concept in manufacturing, yet its definition varies depending on what activities are included in it. Before looking at how lead time can be predicted, it is important to understand what it consists of, why it is difficult to estimate, and why current planning methods fail. This section covers three main topics, the structure of lead time and its components, the differences in lead time between make-to-order and make-to-stock manufacturing environments and the limitations of how lead times are currently managed in ERP systems. Table 1 summarizes definitions of lead time in different sources.

Table 1 Definitions of lead time in manufacturing literature

Source	**Term**	**Definition**
Kingsman, Tatsiopoulos & Hendry (1989)	Total manufacturing lead time	The time elapsed between arrival of raw material and the delivery date. Composed of pool delay plus shop floor throughput time.
Lingitz et al. (2018)	Manufacturing lead time	The time a product needs to pass through the manufacturing process from release to completion.
This study	Production lead time	The number of business days between the actual production release date and the actual production finish date. Includes component procurement, assembly, and routine testing. Excludes extended testing, packing, and modifications.

Manufacturing lead time is not just one single activity but rather a chain of time components that add up to the total duration. Kingsman et al. (1989) break this down into a clear hierarchy. At the lowest level is the operations lead time, which includes setup time, actual processing, transportation and waiting in queue at each work centre. The shop floor throughput time is the time elapsed between an order onto the shop floor and its completion, covering all operations the order must pass through. Before the order reaches the shop floor, it typically sits in a pool of unreleased jobs. This pool delay plus the shop floor throughput time together makes up the total manufacturing lead time. The total delivery lead time includes also the quotation time, design time and material lead time.

An important finding from this is the actual processing time is usually only a small part of the total lead time. Most of the time is spent waiting. Kingsman et al. (1989, p. 197) note that manufacturing lead times in make-to-order companies are “often long and

unreliable yet only a small proportion is due to the actual processing time." A related pattern was found nearly thirty years later in a case company specialized in semiconductor manufacturing (Lingitz et al., 2018). Their study showed that traditional planning methods, which typically rely on historical averages to estimate lead times, produce unreliable schedules because they fail to consider many factors that contribute to variations in delivery times.

The type of manufacturing environment plays a major role in how predictable lead times are. Köber & Heinecke (2012) explain how the make-to-stock (MTS) approach offers properties like high capacity utilization, high availability and short lead times. Make-to-order (MTO) manufacturing on the other hand provides more flexibility and responsiveness to changes in demand and product variability but comes with longer and less reliable lead times. In MTS environments production follows standardized processes for known products which make the lead times stable and reliable. In MTO environments each order can require different combinations of materials, processing steps and testing which creates variability that simple averages from historical data cannot capture. MTO production planning is described as complex because of the high variability in routings, processing times and irregular arrival times of customer orders (Kingsman et al., 1989). The study by Köber & Heinecke (2012) confirm that, in practice, it is difficult to achieve the desired performance using only an MTO strategy, as production capacity and the ability to adapt to changes in demand are limited and costly.

The way lead times are managed in ERP systems makes this variability particularly difficult to handle. Most ERP systems are built on Material Requirements Planning (MRP) which assumes that lead times are static and do not change throughout the planning period (Ioannou & Dimitriou, 2012). Ioannou & Dimitriou (2012) argue that this is the main limitation that makes MRP unsuitable for make-to-order manufacturing environments, because fixed lead times do not reflect the actual workload of production system. When planned lead times are set too tight, orders are late and require expediting. When they are set too loose, overall planning performance suffers. Jodlbauer & Strasser (2019)

identify further weaknesses, noting that MRP also treats production resources as if they had unlimited capacity and uses planned lead times that do not consider for the current workload. They point out that this frequently results in unrealistic production plans that need significant manual correction before they can be executed.

Product customization makes the fixed lead time assumption even more problematic. When products can be configured with many different options, the number of possible combinations grows quickly, and each combination can have different effects on how long production takes. Fixed planned lead times incorrectly assume all configurations take the same amount of time, ignoring the fact that actual lead times can vary depending on the level of customization. This means that even if the fixed lead time were accurate on average for certain product type, it could still overestimate the lead time for simple configurations and underestimate for more complex ones.

The result is a consistent discrepancy between planned and actual lead times. Lingitz et al. (2018) confirm this discrepancy in their semiconductor manufacturing study, where they found that traditional planning methods that rely on average lead times do not capture the real factors that drive lead time variation. Ioannou & Dimitriou (2012) demonstrate that dynamically updating lead time estimates based on actual shop floor workload provides significantly better results than static MRP estimates. Kingsman et al. (1989) observe that customers value dependable delivery commitments and that this reliability can influence which manufacturer they chose to order from, which means that the gap between planned and actual lead times has direct competitive consequences. The disconnect between what ERP systems assume and what happens on the shop floor creates a clear need for methods that can learn the complex relationships between product configurations and production lead time from data. Machine learning has an ability to find nonlinear patterns from large amounts of data which offer an approach to this problem. How machine learning has been applied to lead time prediction is covered in chapter 2.2.

In this study, the production lead time is defined as the number of business days between the actual production release date and the actual production finish date. In the hierarchy defined by Kingsman et al. (1989) this resembles most to the shop floor throughput time. It covers the core manufacturing duration, including procurement of standard components, the assembly process and routine testing. Business days are considered instead of basic calendar days because production does not take place at weekends. The definition excludes on purpose some of the production activities such as special testing, packing and post-production modifications. The time spent on these activities varies considerably and the activities are not part of this study's focus, which is how different product configurations affect the production lead time.

2.2 Machine Learning in Manufacturing

With the rise of Industry 4.0, manufacturing companies have gained access to more data than ever before. At the same time computing power and storing data have become cheaper and more accessible. This combination has made machine learning a practical tool for solving several problems in manufacturing. Usuga Cadavid et al. (2020) reviewed 93 research articles on how machine learning has been used in production planning and control. They found applications in areas such as scheduling, time estimation, quality control, process monitoring and maintenance. The most common approach was supervised learning, and the most popular algorithms were neural networks, decision trees and ensemble methods like Random Forest. Most studies used data from ERP systems or Manufacturing Execution Systems (MES). Despite the growing number of different ML applications, the review found that most of the possible application areas had little or no research attention.

Fahle et al. (2020) came to similar conclusions in their review of ML applications in factory environments between 2015 and 2020. Neural networks and tree-based algorithms were the most common techniques used in areas such as process planning, quality control, maintenance and logistics. Their review only included studies where a machine

learning method had been implemented, not just proposed. Both reviews show that while ML is spreading in manufacturing context, its use for predicting production lead times is still relatively rare compared to other application areas like quality and maintenance.

Estimating manufacturing lead times has long been a familiar concept, but the use of advanced data mining techniques for this purpose was a relatively new approach in the 2000s. A study by Öztürk et al. (2006) used regression trees to estimate lead times in a simulated make-to-order production unit. They started with 26 different attributes describing the job and the shop and then narrowed these down to a small set that worked best for prediction. The regression tree approaches linear regression and three other methods from earlier literature. The most useful attributes turned out to be the total processing time of the order, the number of parts waiting at the first machine and the expected workload on the machines the order would pass through (Öztürk et al., 2006).

When real production data is not available, simulation can be used to generate training data for ML models. In a study by Pfeiffer et al. (2016) they built a simulation model that produced log data similar to what a real MES system would record. They trained decision trees, linear regression and Random Forest models on this data. Random Forest gave the best results. The study showed that the number of jobs in the system with work-in-progress status and the lengths of machine queues were strongly connected to how long production took.

Machine learning based lead time prediction was tested in a real semiconductor manufacturing setting (Lingitz et al., 2018). Working with a semiconductor manufacturer they compared eleven different algorithms including neural networks, support vector machines and Random Forest. The data came from the company's manufacturing execution system (MES). Random Forest produced most accurate predictions. The study showed that ML models trained on real historical production data can predict lead times much

better than traditional methods that simply rely on averages from past data (Lingitz et al., 2018).

Gyulai et al. (2018) explored how ML predictions can support real-time production control decisions. Using a simulation model of a flow-shop system as a testbed, they trained a Random Forest model on both job features like product type and priority and the current state of the production system such as the number of jobs currently being processed. When the model predicted that an order would be late, it automatically raised its priority so it would move through the system faster. This approach reduced overall lateness compared to both a simple first-in-first-out rule and predictions based on Little's law. The prediction accuracy was around 10% normalized root mean square error.

Burggräf et al. (2020) provided the most comprehensive overview of the field by reviewing 42 publications on lead time prediction in make-to-order production. They classified each study by what data is used, where the data came from and what prediction method it applied. Two findings stand out, first almost all studies used order data and information about the production system's current state, but very few used product-specific data such as material properties or dimensions. Material data appeared in only 5% of the publications. Second, as models became more complex by including more types of data, fewer of them used real production data. Most relied on simulations instead. The authors pointed out that this is a gap that needs to be addressed with future research (Burggräf et al., 2020).

A recent study by Rokoss et al. (2024) predicted delivery times for two small make-to-order manufacturers in Germany using real production data from over 16,000 orders. They compared several algorithms and found that XGBoost performed best. An important part of their approach was creating features based on production management knowledge, such as calculating the current workload and identifying bottleneck machines. These features improved prediction accuracy considerably compared to using only the raw order data. They also used SHAP values to measure how much each feature

affected the prediction. The customer's desired delivery date turned out to be the most important single feature, accounting for about one third of the model's output. The study used a time-based validation approach where the training data always came from an earlier period than the test data which prevents the model from accidentally learning from future information (Rokoss et al., 2024).

Several patterns emerge from this review of prior research on machine learning in manufacturing. Most ML-based lead time prediction studies have been using simulated production environments. Very few have focused on make-to-order assembly of configurable products where variety creates a different kind of challenge for prediction. Product configuration data has been largely overlooked as a predictive feature. Burggräf et al. (2020) found that material and product data was used in only 5% of the studies they reviewed. Few studies apply explainability methods like SHAP to understand what drives the predictions and in which direction. Proper temporal validation is handled inconsistently across the literature, and direct comparison of ML predictions against a company's existing ERP planned lead times is rare.

This study addresses several of these gaps. RQ1 tests on whether machine learning models trained on real production data from a make-to-order environment can predict lead time effectively, using approximately 200,000 rows of historical data from the case company instead of simulated data. RQ2 compares three tree-based algorithms to determine which performs best in this context. RQ3 uses SHAP analysis to identify which product configurations most influence lead time predictions, directly addressing the finding by Burggräf et al. (2020) that product-specific data has been largely overlooked. Additionally, the models are evaluated against the case company's existing ERP planned lead times and validated using time-series cross-validation that preserves chronological order.

2.3 Machine Learning Algorithms

Machine learning algorithms used for regression on tabular data come from several families such as linear models, tree-based ensembles, neural networks and support vector machines. Not all of these are equally suitable for the problem in this study, so this section explains which family was selected and why and then describes the three specific algorithms that were used.

Tree-based ensemble methods are widely used in machine learning when working with complex datasets with multiple features. Other algorithm families such as neural networks and support vector machines have also been applied to lead time prediction in manufacturing. For example, Lingitz et al. (2018) tested eleven different algorithms including neural networks and support vector machines, and Burggräf et al. (2020) found that artificial neural networks and regression models were the most common ML approaches for predicting lead times. However, tree-based ensemble methods were selected for this study for the following reasons. First, Lingitz et al. (2018) found that ensemble tree-based models outperformed all other tested methods for predicting manufacturing lead times, with Random Forest achieving the lowest prediction error and a boosted regression tree ranking second. Second, Shwartz-Ziv & Armon (2022) showed that tree-based models perform competitively with deep learning on structured tabular data, which is the type of data used in this study. Third, SHAP's TreeExplainer (Lundberg et al., 2020) enables exact and efficient computation of feature importance values for tree-based models, which is essential for addressing RQ3. For these reasons, neural networks and other approaches were not included in the comparison.

2.3.1 Random Forest (RF)

Random Forest, introduced by Breiman (2001), is a popular machine learning method used for predicting categories (classification) or numbers (regression). Instead of relying

on just one decision tree, which easily memorizes training data and performs poorly on new data, Random Forest builds a so called “forest” of many different decision trees.

The model makes these trees different from each other using two types of randomness. First, it uses “bagging” which means each tree is trained on a random sample of data, leaving some data out. Second, at each split in the tree, it only looks at random subset of the features in the dataset.

To get the final prediction in a regression task, the Random Forest simply averages the predictions from all the individual trees from the forest. The formula for it is:

$$\hat{y} = \frac{1}{K}\sum_{k=1}^{K} f_k(x)$$

Where, $\hat{y}$ represents the final prediction, while K is the total number of trees in the forest. The term $f_k(x)$ stands for the prediction made by a single tree. The $\sum$ symbol simply indicates that we add up all the predictions from every single tree and dividing by K gives us the average.

Averaging reduces the variance of the prediction compared to a single tree, which is the main reason Random Forest generalizes better than an individual decision tree on unseen data (Breiman, 2001).

2.3.2 Extreme Gradient Boosting (XGBoost)

While Random Forest builds all its trees at the same time independently, boosting models build trees one after another. Extreme Gradient Boosting (XGBoost), created by Chen & Guestrin (2016), is a very powerful and fast version of this. In XGBoost, each new tree is built specifically to fix the mistakes made by the trees that came before it.

One of the main reasons XGBoost works so well is that it actively penalizes the model if it gets too complicated. This is done with an objective function, which scores how well the model is doing. To calculate this quickly, XGBoost uses Taylor approximation. The formula for a step (t) looks like this:

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} \left[l\left(y_i, \hat{y_i}^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

First, the large, bracketed part measures the error, showing how far off the model's predictions are from the actual values. The g_i and h_i are mathematical terms called gradients that tell the model which direction to adjust to fix its errors. Secondly, the $\Omega(f_t)$ part is the penalty or the regularization term. It essentially tells the model to keep the tree as simple as possible. If a tree has too many leaves or the weights are too large, this part of the formula adds a penalty. By balancing the error and the penalty, XGBoost learns highly accurate patterns without overfitting to the training data (Chen & Guestrin, 2016).

2.3.3 Light Gradient Boosting Machine (LightGBM)

Even though XGBoost is fast, it can still take a long time to train if you have millions of rows of data, because it must look at every single possibly split for every feature. To solve this problem, Ke et al. (2017) introduced LightGBM, which is designed to be extremely fast and use less memory.

LightGBM speeds things up in a few ways. Instead of looking at every exact number to make a tree split, it groups continuous numbers into bins, like a histogram. It also uses a technique called Gradient-based One-Side Sampling (GOSS), which basically ignores the data points the model is already predicting correctly and focuses only on the data points it is struggling with.

The biggest difference between LightGBM compared to Random Forest and XGBoost, is how the trees grow. Random Forest and XGBoost grow trees level-wise, meaning they build the tree one full layer at a time. LightGBM grows its tree leaf-wise. It looks for the single leaf that will improve the model the most and splits that one, regardless of which level it is on.

Leaf-wise growth tends to be faster and often more accurate than level-wise growth, but it can build deep and unbalanced trees. This makes LightGBM prone to experience overfitting if you are working with a small dataset which is why it is always necessary to limit the maximum depth of the tree when setting up the model (Ke et al., 2017).

2.4 Feature Engineering

Feature engineering is a critical component in the machine learning pipeline. It serves as a bridge between raw data and predictive models. Feature engineering is the transformation of raw data to a format that machine learning algorithms can understand. This process increases the model's performance significantly. Researchers report that most of the time used in developing machine learning models is used in feature engineering and data preparation (Zheng & Casari, 2018 p. vii.) Verdonck et al. (2024 p. 3918) define feature engineering as the transformation, extraction and selection of the most relevant features to improve the performance of the machine learning model. This process helps the algorithms to find patterns from data with many redundant features.

Santoso & Priyadi (2024 pp. 418-419) state that feature engineering helps machine learning algorithms to capture non-linear relationships from data which might not be visible for them in raw data form and that poorly processed data can cause serious problems for the model's performance and generalizability. Understanding how different feature engineering methods work, can optimize the learning process and lead to more accurate models. Feature engineering helps to transform raw data into representations

that highlight relevant patterns for the machine learning algorithms while also reducing noise present in raw data (Santoso & Priyadi, 2024 p. 419.)

Machine learning algorithms understand usually numerical features better than categorial. A categorial feature or variable consists of a set of categories and is considered a qualitative variable whereas numerical variables are quantitative in nature (Valdez-Valenzuela et al., 2021. p. 2) For this reason, categorical variables often require additional encoding for machine learning algorithms to properly process them since most algorithms expect numerical inputs. Zheng and Casari (2018 pp. 78-79) identify one-hot encoding as the most common solution for this problem where categorical variables with k categories are transformed into multiple binary indicator variables, each representing the presence or absence of a category. For example, a color variable with categories (red, green, blue) becomes three binary variables: is_red, is_green, is_blue as seen in Figure 1.

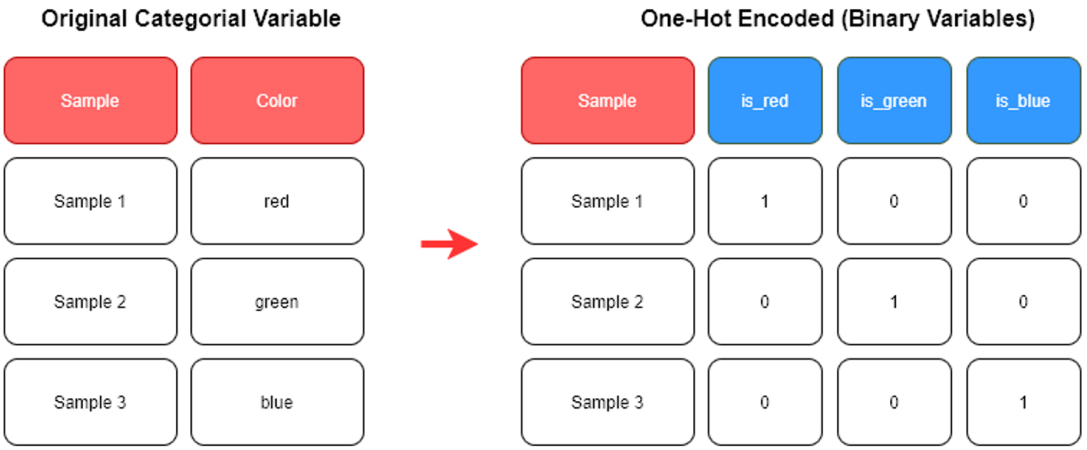

Figure 1. Example of One-Hot Encoding.

While one-hot encoding is widely used, other encoding methods exist also and are used depending on the categorial variable type and the algorithm used. Although it may seem intuitive to assign each category a unique integer value (e.g., red=0, green=1, blue=2), Zheng & Casari (2018 p. 78) caution against this approach because “the resulting values would be orderable against each other, which should not be permissible for categories”.

Instead, they recommend dummy coding, which uses k-1 features to represent k categories, eliminating one category as a reference while maintaining interpretability as seen in Figure 2.

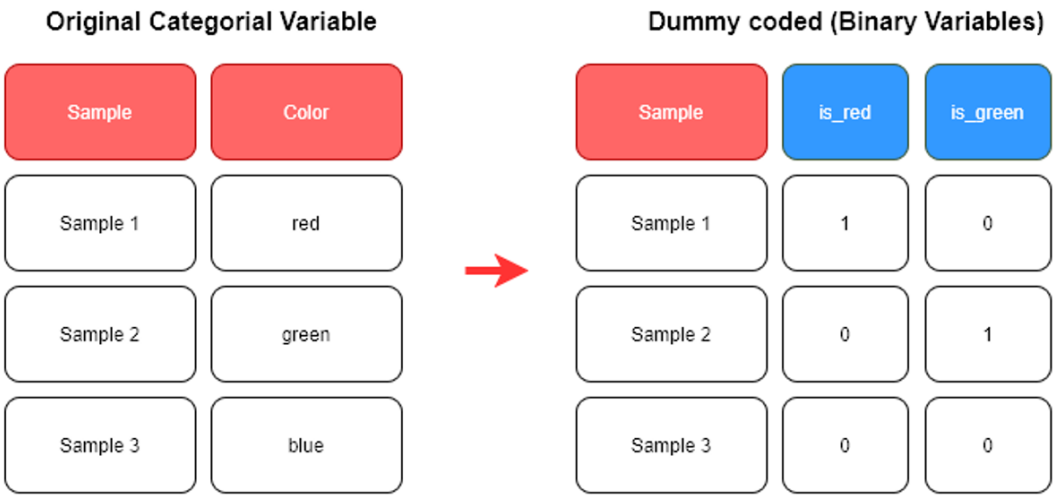

Figure 2. Example of Dummy Coding.

For categorical variables with many categories, different encoding strategies are in place depending on the objectives of the study. Zheng & Casari (2018 p. 95) explain, that tree-based models need to perform repeated searches over all features to find optimal splits. This makes high-dimensional encodings such as one-hot encoding with hundreds of categories computationally expensive for the machine learning algorithms because each binary column created must be checked as a potential split point. Even though the computing time is increased with one-hot encoding, the research by Pargent et al. (2022, p. 2686) shows that tree-based models such as Random Forest and XGBoost perform better with one-hot encoding compared to dummy coding when dealing with high-dimensional data.

2.5 Explainable Artificial Intelligence

Complex machine learning models can achieve high predictive accuracy, however their internal logic is difficult to follow, and it is often unclear why a specific prediction was made. This lack of transparency is recognized well in literature and Barredo Arrieta et al. (2020) note that increasing model complexity tends to reduce interpretability. This makes models harder to understand and trust in practice.

Explainable artificial intelligence (XAI) methods aim to address this problem. They can be divided into two broad categories. Transparent models are interpretable by design meaning their internal logic can be read directly without additional tools. Post-hoc explainability methods are applied after training to explain models that are not inherently interpretable. These include feature relevance methods, local explanations, explanations by simplification and visualization techniques (Barredo Arrieta et al., 2020). Tree ensembles fall into the black-box category and require post-hoc methods.

A further difference is made between global and local explanations. Local methods explain individual predictions, whereas other approaches aim to summarize model behaviour globally (Barredo Arrieta et al., 2020). LIME (Local Interpretable Model-agnostic Explanations) explains individual predictions by fitting a simple linear model around a single instance it can be applied to any black-box classifier (Ribeiro et al., 2016). It does not describe the model, only the area around one specific prediction.

SHAP (SHapley Additive exPlanations) is a feature relevance method based on cooperative game theory. It assigns each feature an importance score that reflects its contribution to a specific prediction. Lundberg & Lee (2017) showed that SHAP is the only additive attribution method satisfying three properties, local accuracy, missingness and consistency simultaneously. These properties mean that the scores are faithful to the model and that more influential features receive higher values.

For tree-based models, Lundberg et al. (2020) developed TreeExplainer, which computes exact SHAP values efficiently. This makes SHAP practical for large datasets and complex ensembles such as XGBoost and Random Forest. Lundberg et al. (2020) also note that tree-based models perform particularly well on structured tabular data, including data from industrial environments.

In this thesis, SHAP is applied using TreeExplainer to identify which features most influence lead time predictions. This addresses the third research question which asks what factors influence lead time predictions and whether the model behavior aligns with manufacturing domain knowledge.

3 Methodology

This chapter explains the chosen methodological approach of the thesis. Several research approaches were considered for this study. Design Science Research (DSR) was selected because the primary goal of this study is to build a new artifact and evaluate whether it solves a practical problem. DSR provides a structured process for creating the machine learning model as an artifact and evaluating it against clearly defined criteria, which directly supports the research questions.

The study follows the Design Science Research methodology to develop and validate an artifact which in this study is a machine learning model for predicting production lead time in a make-to-order environment. The chapter is organized into four sections: the DSR process and how it is applied in this study, the case company and data context, the extraction of historical production data, and the criteria used for evaluating the developed artifact.

3.1 Design Science Research Process

This study follows the Design Science Research (DSR) paradigm, which fits well when the research aims to solve a practical problem through the creation and evaluation of an artifact. In Design Science Research, an artifact can be a model, method, construct or instantiation that addresses a real-world problem (Hevner et al., 2004). In this study, the artifact is a machine learning prediction model that estimates production lead times based on ERP data.

The research process follows the six-step Design Science Research Methodology (DSRM) proposed by Peffers et al. (2007). The DSRM provides a widely accepted framework for conducting design science research in information systems, incorporating principles, practices, and procedures required to carry out research. Figure 3 illustrates how the DSRM process is applied in this study, and the steps are described in detail below.

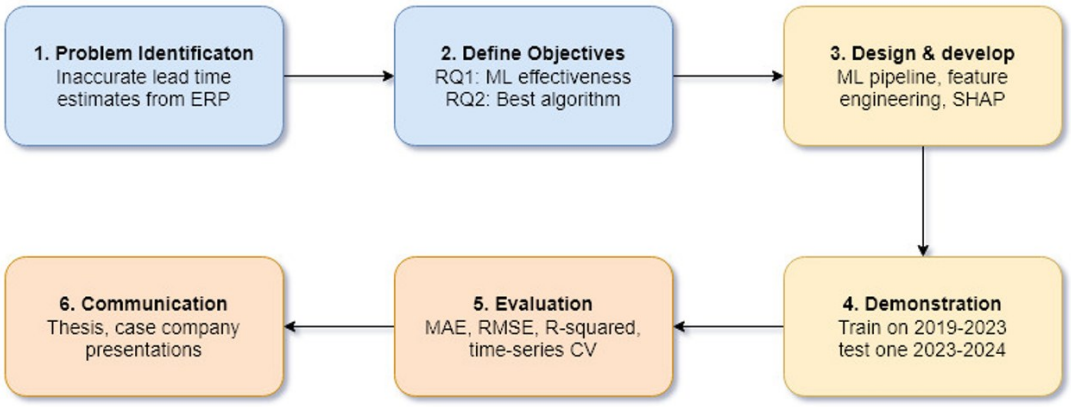

Figure 3. Design Science Research Methodology applied in this study (adapted from Peffers et al., 2007).

The first step, problem identification and motivation, identifies the practical problem faced by the case company. The company produces electric motors with numerous product configurations, and the current ERP-based planning methods struggle to estimate production lead times accurately for orders with varying combinations of features. This leads to unreliable delivery commitments and problems in production planning. The problem is described in the introduction and supported by the literature review.

The second step, defining objectives for a solution, translates the practical problem into three research questions. RQ1 asks how effectively machine learning models can predict lead time using ERP data. RQ2 asks which algorithm provides the most accurate predictions. RQ3 asks which product configurations have the most significant impact on production lead time.

The third step, design and development, is where the machine learning pipeline is built. This includes data cleaning, feature engineering from product descriptions and variant codes, temporal train-test splitting, training three tree-based ensemble models (Random Forest, XGBoost and LightGBM) and finally applying SHAP analysis for model interpretability.

The fourth step, demonstration, applies the developed models to the case company's historical production data from the ERP system. The models are trained on the production orders dated from 2019 to middle of 2023 and tested on orders from middle of 2023 to the end of 2024. This simulates how the models would perform in the future if applied for unseen orders.

The fifth step, evaluation, compares the trained machine learning models' predictions to the case company's existing ERP-based planning dates. The evaluation uses Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R-squared as performance metrics. The machine learning models are compared against two baselines: the case company's ERP-based planned lead times and a naive baseline predicting the training set mean. Time-series cross-validation is used to verify that the results are stable across different time-periods because of the nature of the used production data. The evaluation results are explained in Chapter 6.

The sixth and final step, communication, is realized with this thesis and through presentations made for the case company, especially the project management team.

3.2 Case Company & Data Context

The case company is a manufacturing unit of a large international corporation, located in Vaasa, Finland. The unit specializes in producing low-voltage electric motors that comply with the International Electrotechnical Commission (IEC) standards. These motors are used across a wide range of industrial applications, including pumps, fans, compressors and conveyors.

The company operates in a make-to-order (MTO) manufacturing environment, where each motor can be configured with various options to fit customer requirements. A production order typically consists of a base motor type, frame size and pole pair, combined with a set of variant codes the specify additional features required by the customer. This

in mind, the number of possible product configurations is extremely large, which makes lead time estimation considerably more challenging than in standard make-to-stock environments.

The company uses SAP as its Enterprise Resource Planning (ERP) system for managing the production orders. The ERP system records planned and actual dates for key production milestones, as well as product specifications including the motor description and variant codes applied in each order. Production planning has relied for a long time and still relies on predefined standard lead time for different product categories, with some product configuration exceptions that add more time to the lead time of the order, however this is now considered outdated and needs to be updated to meet modern industry standards of an MTO environment.

3.3 Extraction of Historical Production Data

The data used in this study was extracted from the company's SAP system. The extraction covered production orders from 2019 to 2024 and resulted in a dataset of over 200,000 rows of data. Each row represents one production order and contains the necessary features such as product description, frame size, pole pairs, planned and actual production release dates, planned and actual production finish dates and a comma-separated list of variant codes applied to that specific order. The data was extracted as an Excel export from SAP.

The production lead time, which serves as the target variable for the machine learning models, is calculated as the number of business days between the actual release date and the actual production finish date. Business days are used instead of calendar days, since the factory's production does not occur on the weekends usually. The planned lead time is calculated the same way from the planned production release and finish dates, which serves as a baseline for comparison but is excluded from the input features of the machine learning models. The reason behind this exclusion is that one of the goals of

this study is to understand how product configurations affect the production lead time, not to assess how well the ERP system's own estimates serve as a feature to predict the actual lead time.

The data used in this study was validated through discussions with the case company's Project and Delivery Support Manager in several meetings. These meetings confirmed the features wanted investigated and helped to define the scope of the study. The scope was limited only to the M3-series motors, which represent the relevant product generation manufactured at the production unit.

3.4 Artifact Evaluation Criteria

The developed artifact is evaluated against both the quantitative performance and practical relevance criteria, consistent with the DSR evaluation guidelines defined by Hevner et al. (2004).

The primary quantitative evaluation metric is Mean Absolute Error (MAE), which measures the average prediction error in business days. MAE is chosen as the main metric because it is the most intuitive metric for quantifying average error magnitude (Willmott & Matsuura, 2005). Root Mean Square Error (RMSE) is reported alongside MAE to provide an additional perspective, as RMSE gives more weight to larger prediction errors. R-squared is included as a measure of explained variance and Mean Absolute Percentage Error (MAPE) is included to show the size of the error relative to the actual lead time.

The machine learning models are benchmarked against two baselines. The first baseline is the case company's existing ERP-based planning system, represented by the planned lead times recorded in the SAP data. The second baseline is a naive baseline that always predicts the training set mean lead time. For the comparison to be fair, both baselines and machine learning models are evaluated on the same held-out test consisting of the most recent production orders (middle of 2023 to the end of 2024).

To ensure that the results aren't only based on a single train-test split, time-series cross-validation with five folds is used. Unlike standard k-fold cross-validation, which shuffles data randomly, time-series cross-validation preserves the chronological order of the data. In each fold, the model is trained only on past data and validated on future data. Bergmeir & Benítez (2012) examine what happens when different cross-validation strategies are applied to time series data. Standard cross-validation assumes that data points are independent from each other, which is not the case when the data has temporal dependencies. Their empirical study found no significant practical consequences from these theoretical issues, but the use of cross-validation techniques led to more robust model selection compared to evaluating the model on only a single held-out block of data. They recommend blocked cross-validation as a standard procedure because it addresses theoretical concerns while still making full use of available data. In this study, time-series cross-validation with expanding windows is used, which preserves the chronological order and reflects how the model would be used in practice.

In addition to prediction accuracy, the artifact is evaluated on its interpretability through SHAP (SHapley Additive exPlanations) analysis. SHAP is used to identify which product features contribute most to the predictions and in which direction. This is essential for practical adoption, as production planners need to understand why the model predicts a certain lead time, not only what the prediction is. The SHAP analysis directly supports RQ3 by identifying which product configurations most significantly influence lead time.

Finally, the artifact is evaluated on its practical applicability. A prediction utility function was developed that allows the case company to generate lead time predictions from an Excel file without running the full analysis pipeline. This demonstrates that the artifact can be integrated into the company's existing workflow with minimal effort.

Several measures are taken to ensure the trustworthiness of the results. Internal validity is supported by time-series cross-validation, which prevents information from future

orders leaking into the training data. Using multiple evaluation metrics such as MAE, RMSE, R-squared and MAPE and comparing the models against two baselines reduces the risk of drawing conclusions from a single measure. The machine learning pipeline is built so that the entire process from data processing to model training can be reproduced with the same data which supports the reliability of the results. However, the external validity of this study is limited since the data comes from a single manufacturing unit, so the results may not generalize to other companies, industries or manufacturing environments. This limitation is acknowledged and discussed later in this study.

The evaluation is structured so that each research question is answered by a specific part of the evaluation. RQ1, which asks how effectively machine learning models can predict production lead times, is answered by comparing the prediction error of the trained models against the two baselines on the held-out test set, using MAE as the primary metric and RMSE, R-squared and MAPE as the supporting metrics. RQ2, which asks which algorithm provides the most accurate predictions, is answered by training and tuning all three algorithms on the same data and comparing their performance with the same metrics. RQ3, which asks which product configurations have the most significant impact on production lead time, is answered by applying SHAP analysis to the best tuned model and reporting global feature importance, feature importance by category and the directional effect of individual variant codes and motor families. The rest of this section describes the metrics, baselines, validation strategy and interpretability tools that support the evaluations.

4 Data Processing and Feature Engineering

This chapter describes how the raw production data extracted from SAP was transformed into a dataset suitable for machine learning. The steps are presented in the order they were performed in the analysis pipeline, which is data cleaning, feature engineering, dataset splitting and last definition of baseline models.

4.1 Data Cleaning and Processing

The raw data contained 199,906 production order lines extracted from the case company's SAP system, covering the manufacturing period from January 2019 to December 2024. Each row represents one production order with fields including the basic product description, variant codes, planned and actual production release dates and planned and actual production finish dates.

The target variable is the actual production lead time which was calculated as the number of business days between the actual production release date and the actual production finish date. Business days were used because production does not take place on weekends in the case company, so calendar days would overestimate the actual production duration. The calculation was implemented using NumPy's busday_count function, which counts only weekday dates between two dates:

```
def calculate_business_days(start_date, end_date):
     mask = start_date.notna() & end_date.notna()
    result = pd.Series(np.nan, index=start_date.index)
    result[mask] = np.busday_count(
        start_date[mask].values.astype('datetime64[D]'),
        end_date[mask].values.astype('datetime64[D]')
    )
    return result

df['ActualLeadTime'] = calculate_business_days(
    df['Actual release date'], df['MotActFinish']
)
```

The planned lead time was calculated using the same logic from the planned dates and is used later as a baseline for comparison but is not included as an input feature for the machine learning models.

Several cleaning steps were applied to the data. First, 10 rows with missing actual lead time values were removed. Then to filter out potential data errors, the data was checked for negative actual lead times and lead time exceeding the 365-day outlier threshold. Next, the data was filtered to include only M3-series motors, which is the product scope defined for study. Finally, the frame sizes parsed from the product descriptions were validated against the standard IEC frame sizes used for M3-series motors (71, 80, 90, 100, 112, 132, 180, 200, 225, 250, 280, 315, 355, 400, 450 and 500). All orders which had corrupted frame size values caused by unusual formatting in the product description field were removed.

After cleaning, the final dataset contained 199,880 production order lines. Table 2 summarizes the descriptive statistics for the actual and planned lead times.

Table 2. Descriptive statistics of lead times after data cleaning (in business days).

Metric	**Actual lead time**	**Planned lead time**
Mean	26.1	19.8
Median	23.0	19.0
Std	12.7	8.9
Min	2	-224
Max	330	320
Available rows	199,880	199,880

The difference between the planned and actual means indicates that the ERP system's planned lead times tend to underestimate the actual production duration on average. The planned lead time also contains negative values with minimum of -224 business days. This means that some orders have a planned finish date before the planned release date

which is a data quality issue in the ERP system. These records are not filtered out because the planned lead time is not used as an input feature for the machine learning models, but they further illustrate the limitations of the current planning system.

Figures on pages 37 and 38 provides a visual representation of the data after cleaning. The distribution of actual lead times is right skewed with most orders completed within 10 to 40 business days and a long tail of orders extending up to 330 days. The mean of 26.1 days is higher than the median of 23.0 days which is typical for right skewed distributions. The planned versus actual lead time scatter shows a weak relationship between the ERP system's planned lead times and the actual production lead times with a wide spread of points around the diagonal line. The negative planned lead time values are visible on the left side of this plot. The monthly trend shows that average lead times increased approximately 20 days in early 2019 to around 25-30 days in 2021-2023, likely reflecting changes in the production conditions over time. The boxplots by motor family show that the most common families (M3BP, M3GP, M3JP) have similar median lead times around 20-30 days while less common families like M3AA, M3BL and M3EH show higher medians and more variability.

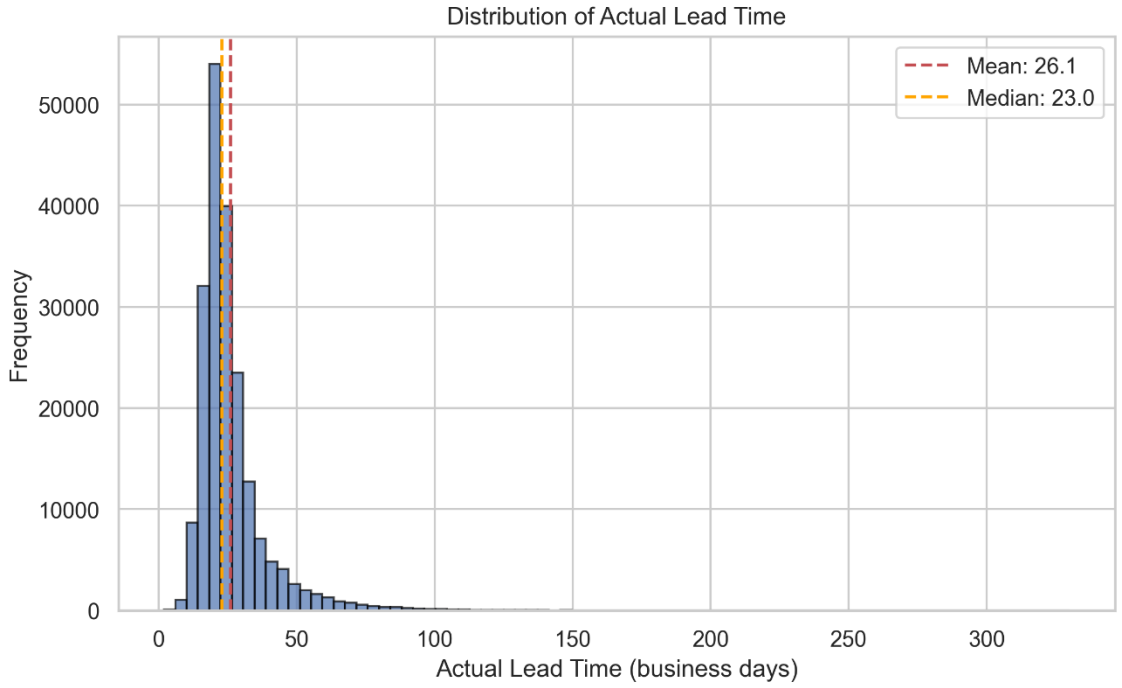

Figure 4 Distribution of Actual Lead Time

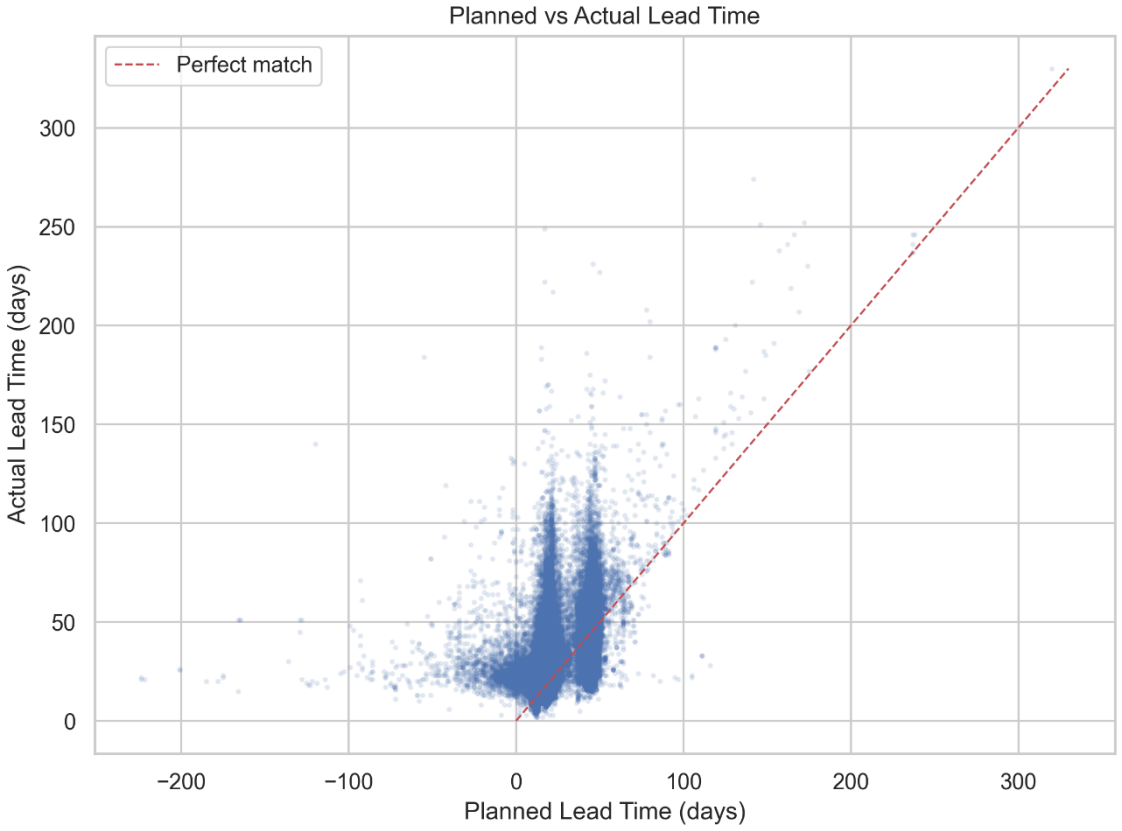

Figure 5 Planned vs Actual Lead Time

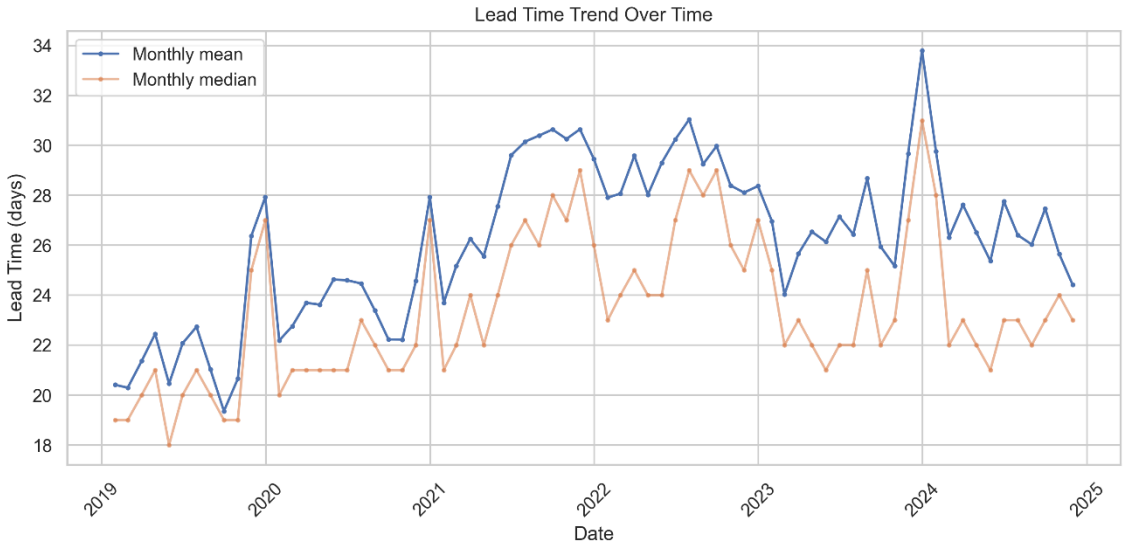

Figure 6 Lead Time Trend Over Time

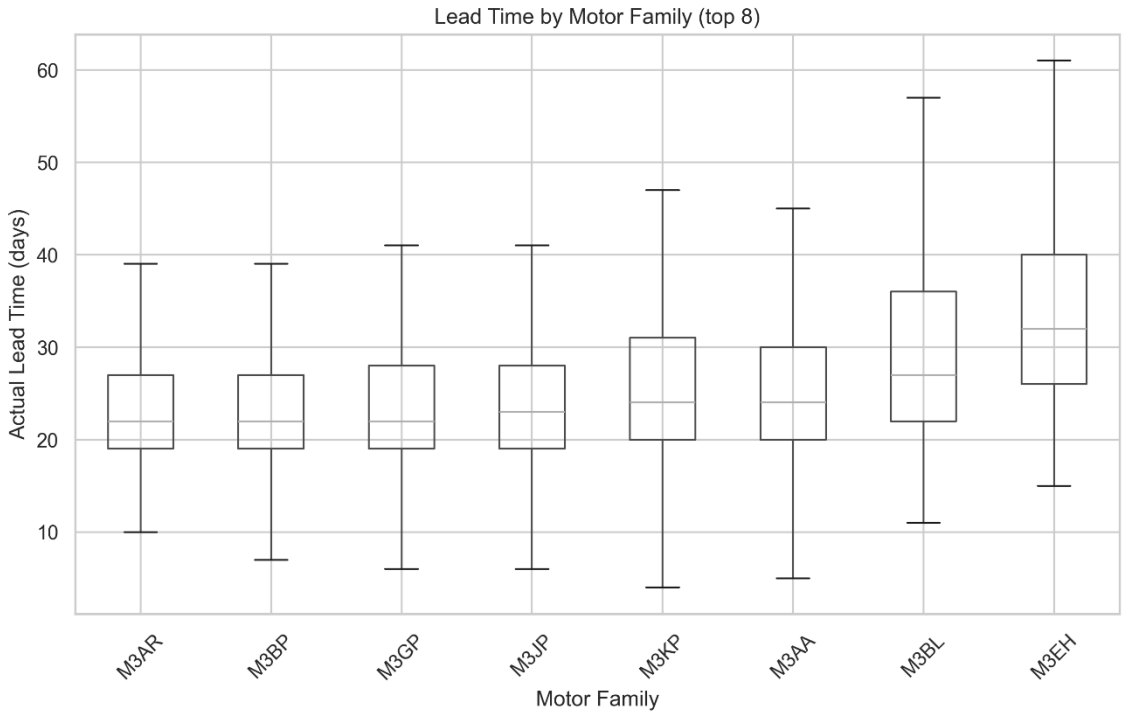

Figure 7 Lead Time by Motor Family

4.2 Feature Engineering

The feature engineering process transformed the raw SAP fields into a structured feature matrix that the machine learning algorithms can process easier. Three types of features were created, features parsed from the basic product description, features parsed from the variant codes' field and a count feature.

The basic description field in the SAP data contains the motor type, frame size and pole pairs in a string format. For example, "M3BP 160MLA 4" describes a motor from the M3BP family, with frame size 160 and 4 pole pairs. This field was parsed into three different features MotorFamily (categorial), FrameSize (numeric) and PolePairs (numeric). The parsing function splits the basic description string and extracts each component with following functions:

```
def parse_basic_description(desc):
    parts = str(desc).split()
    motor_family = parts[0] if len(parts) > 0 else 'Unknown'

    frame_info = parts[1] if len(parts) > 1 else ''
    frame_size = ''.join(c for c in frame_info if c.isdigit())
    frame_size = int(frame_size) if frame_size else 0

    pole_pairs = int(parts[2]) if len(parts) > 2 \
        and parts[2].isdigit() else 0

    return motor_family, frame_size, pole_pairs
```

The dataset contained 30 unique motor families with M3BP being the most common at approximately 101,000 orders followed by M3KP, M3GP and M3AA.

The VariantCodes field contains a comma-separated list of codes that specify additional product features for each order. For example, an order might have the codes "148, 002, 114" which mean three different specific customizations options that a motor can have. Each order had on average 6.8 variant codes with maximum of 36 variant codes on a single order. About 6.1% of orders had no variant codes at all. The variant codes were parsed into a set of individual codes for each order.

The categorial features were encoded using one-hot encoding where each unique category becomes a separate binary column indicating its presence or absence in the order. Each unique variant code was encoded as a binary feature, and each motor family was encoded the same way. As discussed in Section 2.4 of this study, one-hot encoding was chosen because tree-based models have been shown to perform well with this encoding method (Pargent et al., 2022). In addition to the created binary features, a VariantCount feature was created that records the total number of variant codes per order as a single numeric value. This captures the overall level of customization regardless of which specific variant codes are present or absent.

To prevent data leakage, only training data was used to create the binary features of the variant codes and motor families. The following function shows how the feature matrix was constructed with this constraint:

```
def create_features(data, unique_variants, motor_families):
    data = data.copy()
    data['VariantSet'] = data['VariantCodes'].apply(get_var-
iant_set)

    # one-hot encode each variant code seen in training
    for variant in unique_variants:
        data[f'Var_{variant}'] = data['VariantSet'].apply(
            lambda x: 1 if variant in x else 0
        )

    # total number of variants as a numeric feature
    data['VariantCount'] = data['VariantSet'].apply(len)

    # one-hot encode motor family
    for family in motor_families:
        data[f'MotorFamily_{family}'] = \
            (data['MotorFamily'] == family).astype(int)

    motor_family_cols = [f'MotorFamily_{f}' for f in mo-
tor_families]
    variant_cols     = [f'Var_{v}' for v in unique_variants]
    base_cols          = ['FrameSize', 'PolePairs', 'Vari-
antCount']

    feature_cols = motor_family_cols + base_cols + vari-
ant_cols
    return data, feature_cols
```

The unique_variants and motor_families lists are derived from the training set only. When the same function is applied to the test set with the same lists, any variant code or motor family that appears only in the test data are simply assigned as zero values. This prevents data leakage and ensures that the test accurately reflects how the model would behave in a real-world deployment in the case company. The final feature matrix contained 487 features. Table 3 summarizes the feature groups.

Table 3. Summary of engineered features.

Feature group	Count	Type	Example
Motor family	30	Binary	MotorFamily_M3BP
Frame size	1	Numeric	160
Pole pairs	1	Numeric	4
Variant count	1	Numeric	7
Variant codes	454	Binary	Var_148
Total	487		

4.3 Dataset Splitting

The dataset was split into training and test sets using temporal 80/20 split based on the chronological order of actual production release dates. The data was first sorted by the production release date and then the earliest 80% of order lines were assigned to the training set and the most recent 20% to the test set. The 80/20 ration was chosen to balance having enough test data for reliable evaluation while maximizing the training set size for model learning.

```
df = df.sort_values('Actual release date').reset_in-
dex(drop=True)

split_idx = int(len(df) * 0.8)
df_train = df.iloc[:split_idx].copy()
df_test  = df.iloc[split_idx:].copy()
```

The training set contained 159,904 order lines from January 2019 to July 2023, and the test set contained 39,976 order lines from July 2023 to December 2024.

A temporal split was used instead of a random split because the production data has a natural time ordering. As described in Section 3.4, random splitting would allow information from future orders to influence the training process, which does not reflect how the model would be used in practice. With a temporal split the model is always trained

on past data and evaluated on future data which simulates real-world deployment of the model where it must predict lead times for orders it has not seen before.

4.4 Definition of Baseline Models

Two baseline models were defined to provide benchmarks against which the machine learning models are compared. Both baseline models are evaluated on the same held-out test set as the machine learning models to ensure fair comparison.

The first baseline is the case company's existing ERP-based planning system, represented by the planned lead times recorded in the SAP data. For each production order, the ERP system assigns a planned production release date and a planned production finish date. The planned lead time is calculated as the number of business days between these two dates, using the same business day calculation as the actual lead time. This baseline is the most meaningful benchmark because it represents the company's current production planning practice. If the machine learning models cannot outperform the ERP system's estimates, they would not offer practical value for the case company.

The second baseline is a naïve model that always predicts the training set mean lead time for every order, regardless of its product configuration. The mean is calculated from the training set only to avoid any information leakage from the test period. This baseline represents the simplest possible prediction strategy and serves as a lower bar that any reasonable machine learning model should be able to beat. If a machine learning model cannot outperform the naive mean, it has failed to learn any useful patterns from the training data.

The following code shows how both baselines are evaluated on the test set:

```
# ERP baseline — uses the planned lead times from SAP
erp_mask    = ~np.isnan(y_planned_test)
erp_metrics      =      calculate_metrics(y_test[erp_mask],
y_planned_test[erp_mask])

# Naive baseline — always predicts the training set mean
train_mean = y_train.mean()
mean_baseline_metrics = calculate_metrics(
    y_test, np.full_like(y_test, train_mean, dtype=float)
)
```

The ERP baselines are evaluated on all the test rows, and the naive baseline predicts the training set mean of 25.8 business days for every order in the test set. Both baselines are compared against the machine learning models using the same evaluation metrics defined in Section 3.4 including MAE, RMSE, R-squared and MAPE.

5 Artifact Development

This chapter describes how the machine learning artifact was developed. It covers the selection and configuration of algorithms, the validation strategy used to evaluate performance, the hyperparameter optimization process and the setup for model interpretability.

5.1 Algorithm Selection and Configuration

Three tree-based ensemble algorithms were selected for this study, Random Forest, XGBoost and LightGBM. The rationale for selecting these algorithms over other approaches such as neural networks and support vector machines is discussed in Section 2.3. of this study. All three models were first trained using their library default parameters to establish an initial performance comparison. The only parameters set explicitly were the random seed for reproducibility and parallelization settings. The following code shows how the models were initialized and trained:

```
rf_model  = RandomForestRegressor(random_state=42, n_jobs=-
1)
xgb_model = XGBRegressor(random_state=42, n_jobs=-1, verbos-
ity=0)
lgb_model = LGBMRegressor(random_state=42, n_jobs=-1, ver-
bose=-1)

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results[name] = calculate_metrics(y_test, y_pred)
```

The purpose of this initial comparison was to identify the baseline performance of each algorithm before investing computational resources into hyperparameter tuning. All models were trained on the full training set of 159,904 orders.

5.2 Validation Strategy

The artifact was evaluated using a two-level validation approach. The first level is the held-out test set described in Section 4.3. which contains 39,976 orders from July 2023 to December 2024. This test set was not used during any part of the model development process and provides the final performance evaluation reported in this study.

The second level is time-series cross-validation applied to the training data. Five-fold TimeSeriesSplit with expanding windows was used where each fold trains the model on all data up to a certain point and validates on the next chronological block. This means that the training set grows with each hold while the validation set always contains data from a later period than the training data. The following code shows the cross-validation loop:

```
tscv = TimeSeriesSplit(n_splits=5)

for fold, (train_idx, val_idx) in enumer-
ate(tscv.split(X_train), 1):
    X_train_cv, X_val_cv = X_train[train_idx],
X_train[val_idx]
    y_train_cv, y_val_cv = y_train[train_idx],
y_train[val_idx]

    model = _make_model()
    model.fit(X_train_cv, y_train_cv)
    y_pred_cv = model.predict(X_val_cv)

    fold_metrics = calculate_metrics(y_val_cv, y_pred_cv)
    cv_results.append(fold_metrics)
```

Cross-validation serves two purposes here. First, it provides a more stable estimate of model performance than a single train-test split, since results can vary depending on which time period is used for evaluation. Second, it produces confidence intervals that indicate how much the performance varies across different time periods. The confidence intervals were computed using t-distribution.

This validation approach is consistent with the discussion of temporal validation in Section 3.4. The model is always trained on past data and evaluated on future data which prevents information leakage and reflects how the model would be used in practice.

5.3 Hyperparameter Optimization

All three algorithms were tuned using Optuna (Akiba et al., 2019), which uses the Tree-structured Parzen Estimator (TPE) as its default sampling algorithm. Unlike grid search which evaluates all combinations or random search which samples blindly. TPE builds a probabilistic model of objective function and focuses on promising regions of the parameter space which makes it more efficient when the number of hyperparameters is large. Tuning all three algorithms ensures a fair comparison because the default model is not necessarily the best model after optimization.

Each algorithm was tuned with 30 trials and 3-fold time-series cross-validation. The objective function minimized the mean cross-validation MAE. The optimization process differed a bit between the algorithms. For Random Forest, the tuned parameters were n_estimators, max_depth, min_samples_split, min_samples_leaf and max_features. But for XGBoost and LightGBM the tuned parameters included n_estimators, max_depth, learning_rate, subsample, colsample_bytree, reg_alpha and reg_lambda, in addition to those num_leaves was included for LightGBM. The following code shows the Optuna objective function used for tuning XGBoost.

```
def tune_xgboost(trial):
    params = {
        'n_estimators': trial.suggest_int(
            'n_estimators', 100, 400, step=50),
        'max_depth': trial.suggest_int('max_depth', 3, 12),
        'learning_rate': trial.suggest_float(
            'learning_rate', 0.01, 0.3, log=True),
        'subsample':  trial.suggest_float('subsample',  0.6,
1.0),
        'colsample_bytree': trial.suggest_float(
            'colsample_bytree', 0.6, 1.0),
        'reg_alpha': trial.suggest_float(
            'reg_alpha', 1e-8, 10.0, log=True),
        'reg_lambda': trial.suggest_float(
            'reg_lambda', 1e-8, 10.0, log=True),
    }
    model = XGBRegressor(**params, random_state=RANDOM_STATE,
                         n_jobs=-1, verbosity=0)

    mae_scores = []
    for train_idx, val_idx in tscv_tune.split(X_train):
        model.fit(X_train[train_idx], y_train[train_idx])
        preds = model.predict(X_train[val_idx])
        mae_scores.append(
            mean_absolute_error(y_train[val_idx], preds))

    return np.mean(mae_scores)

xgb_study = optuna.create_study(direction='minimize',
    sampler=optuna.samplers.TPESampler(seed=RANDOM_STATE))
xgb_study.optimize(tune_xgboost, n_trials=30
```

5.4 Model Interpretability Setup

SHAP (SHapley Additive exPlanations) was applied to the tuned XGBoost model using TreeExplainer (Lundberg et al., 2020), which computes exact SHAP values for tree-based models. The SHAP values were computed on the training set to provide a good picture of feature importance across the dataset.

The following code shows how the SHAP explainer was set up:

```
import shap

np.random.seed(RANDOM_STATE)
X_shap = X_train

explainer = shap.TreeExplainer(final_model)
shap_values = explainer.shap_values(X_shap)
```

Several types of SHAP outputs were generated to address RQ3 from different perspectives. Global feature importance was computed as the mean absolute SHAP value for each feature, showing which features have the strongest overall influence on predictions. A SHAP summary plot was generated to show both the importance and the direction of each feature's effect on predictions. Dependence plots were created for the most important features to visualize how their values relate to their SHAP contributions. Finally, three SHAP waterfall plots were generated for individual test predictions to illustrate how the model comes to those specific lead time estimates.

Together these analyses provide both a global understanding of what features affect the model's predictions and local explanations of individual predictions which is essential for production planning who need to understand why a specific lead time is predicted not just what the estimated lead time is.

6 Results and Artifact Evaluation

This chapter presents the results of the machine learning artifact and evaluates its performance. The results are organized into three sections, model performance comparison, key factors influencing lead time and case examples of individual predictions.

6.1 Model Performance Comparison

This section compares the prediction accuracy of the three machine learning models against the ERP system baseline and a naive mean baseline. The models are first compared with their library default parameters then after hyperparameter tuning. The best model is further validated with cross-validation, and its prediction errors are analyzed by lead time range, product type and time period.

6.1.1 Default Model Comparison

All three machine learning models outperformed both baselines when trained with their library default parameters. Table 4 shows the performance of each model on the held-out test using four evaluation metrics. MAE (Mean Absolute Error) measures the average prediction error in days. RMSE (Root Mean Square Error) also measures the error in days but gives more weight to large errors. R^2 (coefficient of determination) indicates what proportion of the variance in actual lead times the mode explains where 1.0 would be perfect fit and 0.0 means the model is no better than predicting the average. MAPE (Mean Absolute Percentage Error) expresses the error as a percentage of the actual lead time.

Table 4. Model performance comparison on the held-out test set.

Model	**MAE (days)**	**RMSE (days)**	**R^2**	**MAPE (%)**
ERP Planned	8.12	12.56	0.02	26.23
Global Mean	7.89	12.76	-0.01	26.85
Random Forest (default)	6.84	11.72	0.14	23.30
XGBoost (default)	6.32	11.15	0.22	21.35
LightGBM (default)	6.29	10.87	0.26	21.25
Random Forest (tuned)	6.23	10.96	0.25	20.90
XGBoost (tuned)	6.12	10.55	0.31	20.74
LightGBM (tuned)	6.29	11.06	0.24	21.18

The ERP system's planned lead times produced an MAE of 8.12 days which means the current planning system is off by about 8 days on average. In practical terms and order planned to take 20 days could take anywhere from 12 to 28 days according to the ERP system's accuracy. The naive baseline, which simply predicts the training set average (25.8 days) for every order regardless of its product configuration, performed slightly better with an MAE of 7.89 days. The fact that a completely uninformed baseline performs comparably to the ERP system's planned lead times suggests that the current planning approach captures very little of the order-to-order variation in production lead time. Both baselines had R^2 values near zero which confirm that they explain almost none of the variance of the variance in actual lead times.

All three machine learning models reduced the prediction error substantially compared to the baselines. With library default parameters, LightGBM achieved the lowest MAE of

6.29 days which was followed by XGBoost at 6.32 days and Random Forest at 6.84 days. The two boosting algorithms performed almost identically, while Random Forest lagged a bit behind by about half a day. LightGBM achieved the highest R^2 of 0.26 among the default models, which means that it explains about a quarter of the variance in lead times. Even without any hyperparameter tuning, the machine learning models already reduced the average prediction error by 1-2 days compared to the ERP system.

6.1.2 Tuned Model Performance

All three algorithms were tuned using Optuna with 30 trials each and 3-fold time-series cross-validation. Random Forest showed the largest improvement from tuning, its MAE dropped from 6.84 to 6.23 days, an improvement of 8.9%. XGBoost improved from 6.32 to 6.12 days with only 3.2% improvement, while LightGBM's MAE remained unchanged at 6.29 days. The tuning times differed from each other substantially. Random Forest required 186 minutes for 30 trials due to the computational cost of training large ensembles on 160,000 rows of train data, while XGBoost completed it in 3.7 minutes and LightGBM in 1.5 minutes. Despite Random Forest's large improvement from tuning, XGBoost performed the best with the lowest MAE of 6.12 days and the highest R^2 of 0.31. These results are surprising since LightGBM was the best algorithm with default parameters, but XGBoost won after tuning and Random Forest, which was the weakest with the library settings, came in second with LightGBM performing the worst after tuning.

A possible explanation for why LightGBM did not improve from tuning is that its leaf-wise tree growth combined with the library default settings already produces a model that is well adapted to the structure of this dataset. Leaf-wise growth which is used by LightGBM is more aggressive than level-wise growth used by XGBoost, since it always splits the leaf that has the largest reduction in error rather than expanding the tree symmetrically. The Optuna search with 30 trials did not find a better combination, while XGBoost and Random Forest had more room to improve from their defaults. A search

with more trials might reveal small gains, but the result after 30 trials suggests that LightGBM is already close to its best performance with the library defaults.

Compared to the ERP system, the tuned XGBoost model reduced MAE by 24.7 % from 8.12 to 6.12 days. This means that on average, the model's predictions are 2 days closer to the actual production duration than the ERP system's planned lead times. For production planning scheduling hundreds of orders per month, this improvement helps directly to provide more reliable delivery date commitments and better resource allocation. The R^2 of 0.31 indicates that the model explains about 31% of the variance in actual lead times based only on product configuration data and the feature derived from it. The remaining 69% of the variance is likely driven by factors not available in the input features such as machine availability, workforce levels, supply chain disruptions and other changing conditions in manufacturing. Below figures on pages 53 and 54 show the final comparison of all models as bar chart.

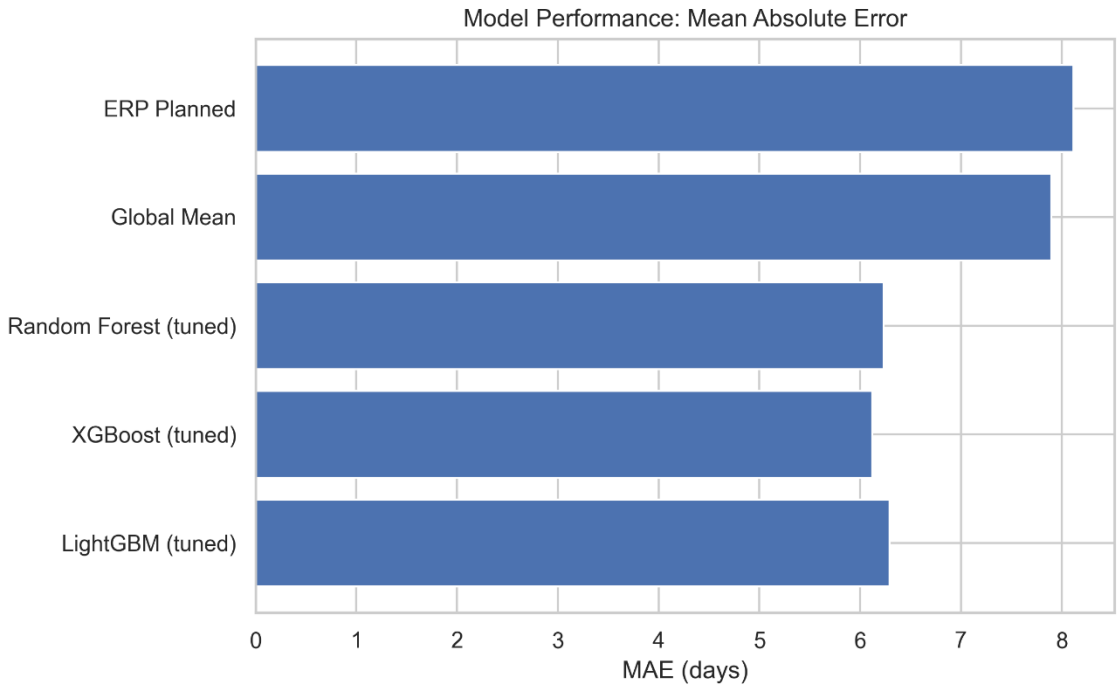

Figure 8 Model Performance (Mean Absolute Error)

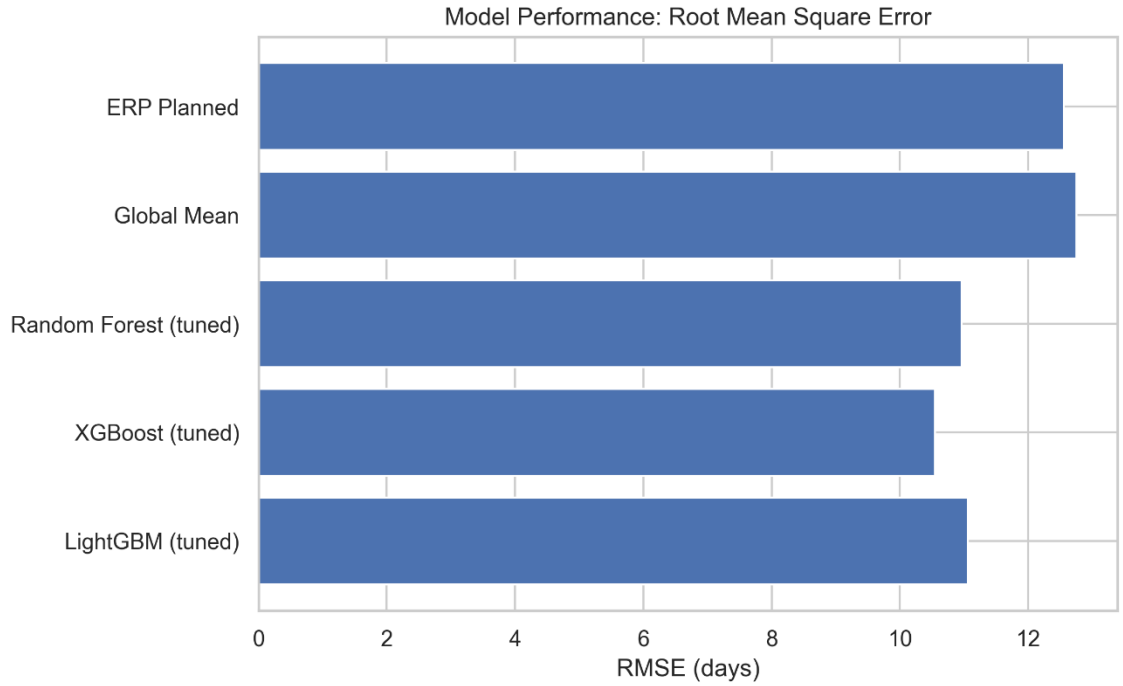

Figure 9 Model Performance (Root Mean Square Error)

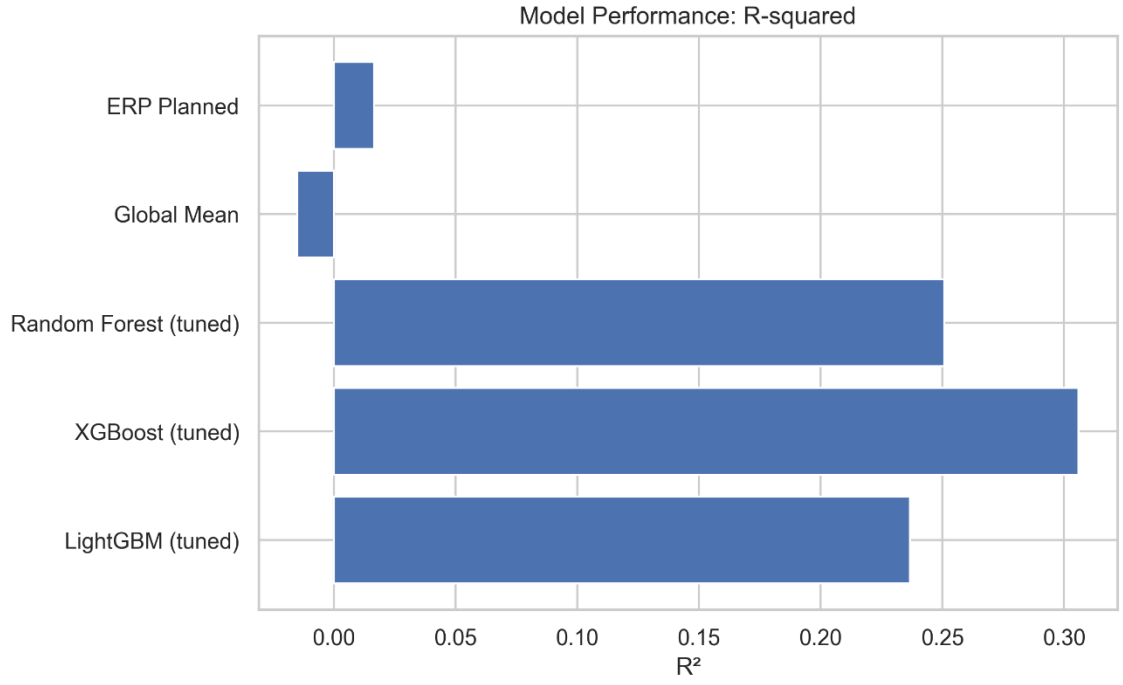

Figure 10 Model Performance (R-squared)

Figure 11 shows the predicted lead time versus the actual lead times for the tuned XGBoost model on the held-out test set. In this scatter plot, each point represents one production order line, and the diagonal dotted line represents how perfect prediction would look like. The model predicts most accurately for orders with actual lead times between 15 and 30 days, where the points cluster near the diagonal. Beyond 30 days, the vertical spread of predictions increases remarkably, for example orders with an actual lead time around 40-50 days receive predictions ranging from 20 to over 100 days. This indicates that the model struggles to distinguish between moderately long and very long orders based on product configuration features alone. For orders with actual lead times above approximately 100 days, the model's predictions appear to reach an upper limit around 80-140 days, meaning it cannot predict lead times beyond this range. This ceiling effect reflects the fact that very long lead times are rare in the training data and are likely driven by rare external features that the product configuration features cannot capture.

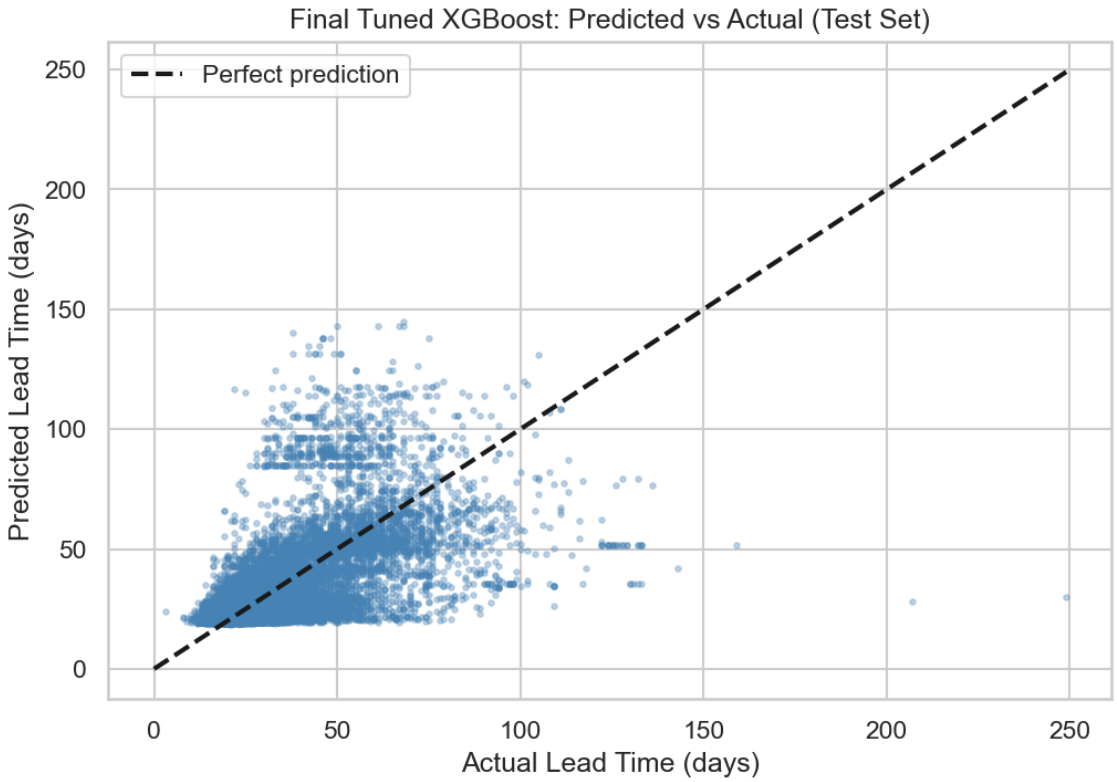

Figure 11. Predicted vs actual lead times (tuned XGBoost model, held-out test set).

6.1.3 Cross-Validation Results

The 5-fold time-series cross-validation with the tuned XGBoost model was performed to verify that the model's performance is not dependent on a single test period. Table 5 shows the results per fold.

Table 5. Cross-validation results per fold (tuned XGBoost model).

Fold	MAE (days)	R^2
1	5.10	0.32
2	4.92	0.32
3	7.97	0.05
4	7.36	0.34
5	5.86	0.49

The cross-validation MAE ranged from 4.92 to7.97 days across five folds. The 95% confidence intervals provide a range within which model's true average performance is expected to fall. For MAE, the interval is 4.55 to 7.93 days with a mean of 6.24 days. For RMSE the interval is 7.20 to 13.59 days with a mean of 10.39 days. For R^2 the interval is 0.11 to 0.50 with a mean of 0.31. These intervals are relatively wide because production lead times behave differently across time periods. The model's accuracy changes over time based on the different types of orders, day-to-day changes in the shop floor and unpredictable events affecting the production.

Fold 3 had notably the worst performance with MAE of 7.97 days and R^2 of 0.05, which indicates a period where the model's learned patterns did not match the actual production. This could be caused by a change in the product mix during that period or by external disruptions such as supply chain issues that affected the lead times in ways the product configuration features cannot capture, which makes the model unprecise in that period. On the other hand, fold 5 achieved the best performance with a MAE of 5.86 days

and R^2 of 0.49 which suggests a period which production behavior was more stable. Despite this variation, the overall cross-validation MAE of 6.24 days is consistent with the held-out test MAE of 6.12 days which confirms that the model generalizes reliably across different time periods.

Figure 12 shows the predicted versus actual lead times for all cross-validation folds combined. The plot shows a dense cluster of points near the diagonal line for orders with actual lead times between 15 and 40 days where the model predicts accurately. For orders with actual lead times above approximately 50 days, the predictions tend to fall below the diagonal, meaning that the model under-predicts the lead time for these longer orders. The model's predictions for these orders are scattered widely, ranging from about 25 to 150 days regardless of whether the actual lead time was 60 or 250 days, which reflects the model's difficulty with orders that deviate far from the typical range.

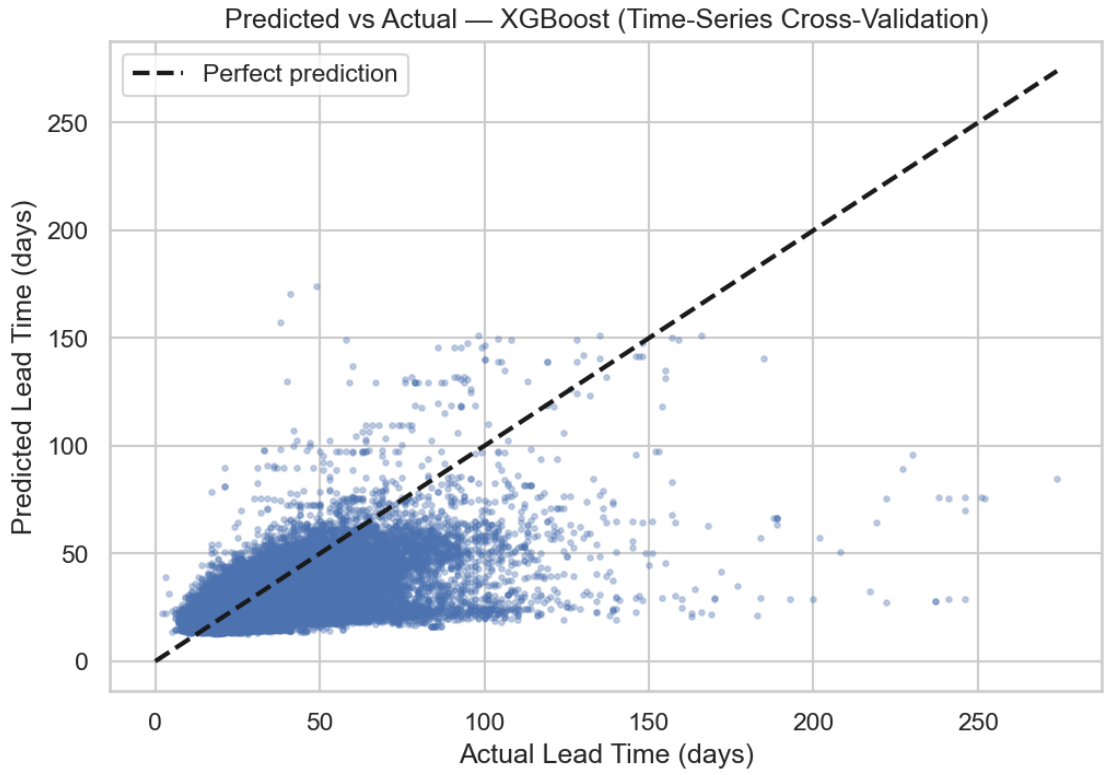

Figure 12. Predicted vs actual lead times (cross-validation).

6.1.4 Residual Analysis

To understand where and how the model makes errors, the prediction residuals were analyzed. In this case the residuals are the difference between the actual and predicted production lead times. The tuned XGBoost model's residuals on the held-out test set have a mean of -0.22 days and a median of -0.99 days, which indicates that the model slightly over-predicts lead times on average but with no meaningful bias. The standard deviation of the residuals is 10.55 days, which indicates substantial variation in individual prediction errors despite the low average error.

Table 6 shows the MAE broken down by actual lead time range to reveal where the model performs well and where it struggles.

Table 6. MAE by actual lead time range (held-out test set).

Lead time range	MAE (days)	Orders (n)	Share of test set
10-20 days	4.80	8,433	21%
20-40 days	4.39	26,990	68%
40-60 days	16.12	3,185	8%
60-365 days	25.10	1,363	3%

The model performs well for orders in the 10–40-day range where it achieves a MAE below 5 days. This range covers 89% of the test set (35,423 out of 39,976 orders), which means that the model is accurate for the majority of production orders the company processes. For orders with lead times above 40 days, the prediction error increases remarkably. Orders in the 40-60-day range have a MAE of 16.12 days and orders above 60 days have a MAE of 25.10 days. This self-explanatory because of two things. First, longer lead times are rarer in the training data, so the model has fewer examples to learn from. Second, extended production durations are more likely to be influenced by factors

outside the product configuration such as production disruptions, material shortages or priority changes during production.

6.1.5 Error by Product Type and Time

The model's accuracy varies considerably across motor families. Table 7 shows the MAE for the ten most common motor families in the test set, ordered by the prediction accuracy.

Table 7. MAE by motor family (top 10 by volume in the test set).

Motor family	MAE (days)	Orders (n)	Average lead time (days)
M3BP	4.95	22,815	25.0
M3JP	5.15	2,366	26.7
M3GP	5.15	4,840	26.5
M3KP	5.68	6,354	30.1
M3AA	8.70	1,009	30.6
M3BL	8.79	1,006	30.4
M3BC	9.10	227	34.5
M3LP	11.81	161	49.2
M3FT	34.85	562	56.2
M3ET	37.92	283	52.3

The model predicts accurately for the most common motor families (M3BP, M3JP, M3GP, M3KP) with MAE values between 4.95 and 5.68 days. These four families account for over 36,000 of the 39,976 test orders (91%), which means the model is practically useful for most of the company's orders. For less common families such as M3AA, M3BL and M3BC, the MAE increases a bit higher than the ERP baseline of 8.12 days.

The highest errors are for motor families M3FT and M3ET where the MAE exceeds 34 days. This is explainable since the motor families are newer than the most common families and their ramp-up has occurred during the time period of the full dataset. The high prediction error is likely explained by the large variability in lead times that when introducing new motor families to the manufacturing process. These results suggest that the model could help production planning in estimating production lead times for the most common motor families while predictions for less frequent motor families should be treated with more caution. Figure 13 and Figure 14 show the MAE by motor family and frame sizes as bar charts.

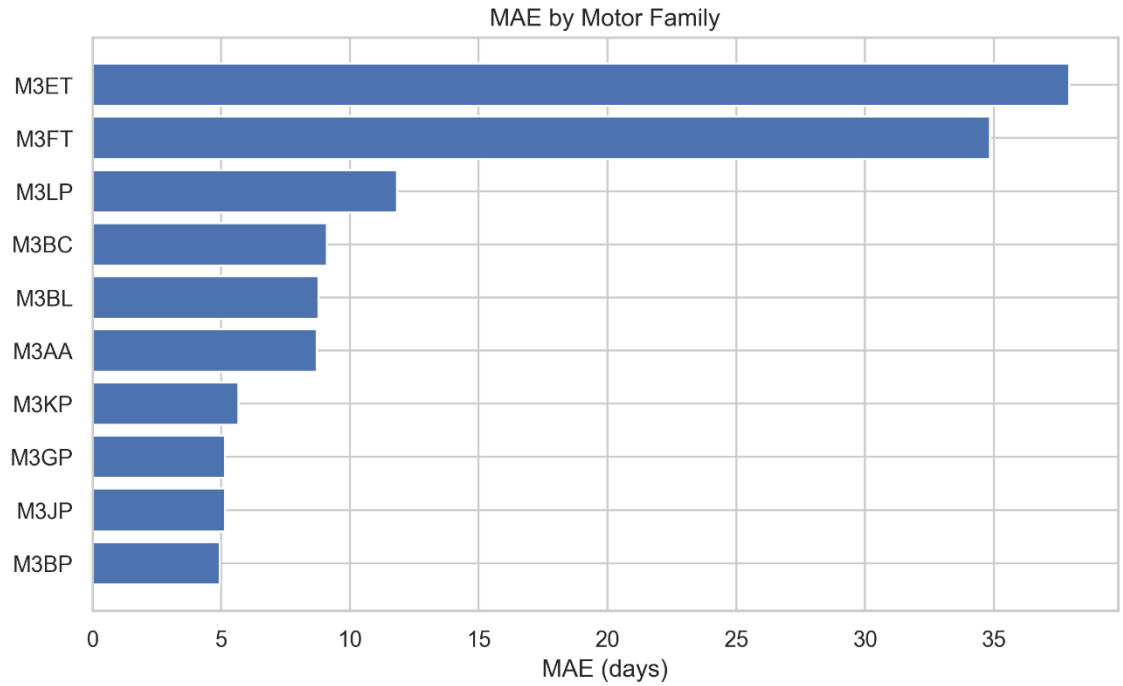

Figure 13 MAE by Motor Family

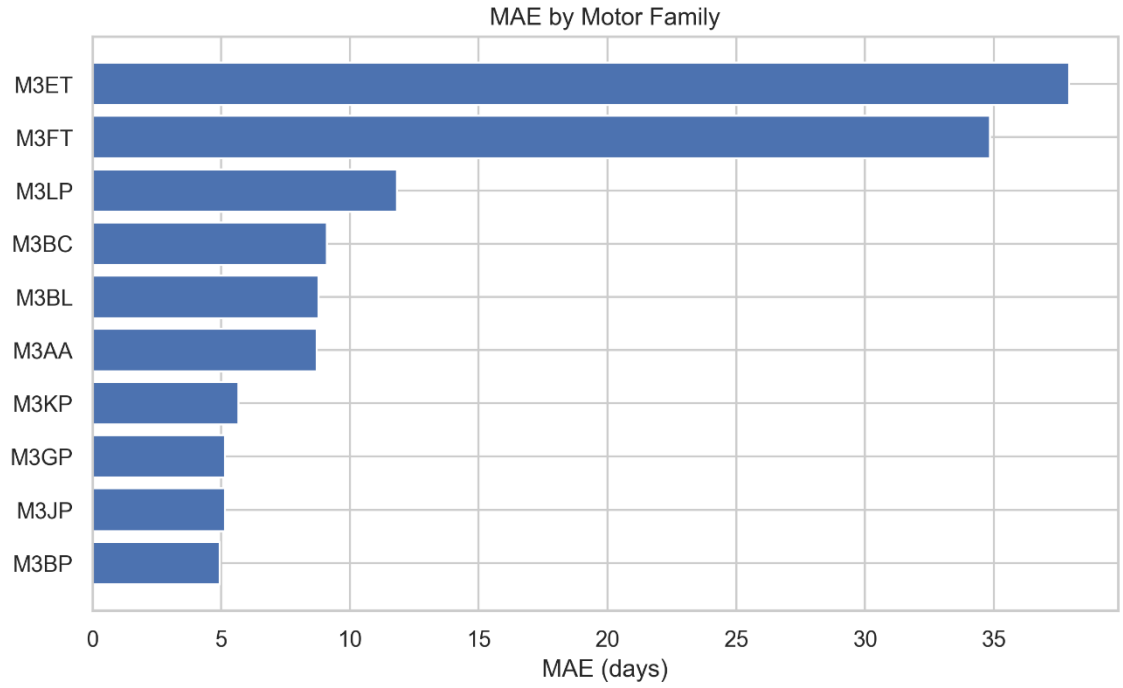

Figure 14 MAE by Frame Size

To evaluate whether the model's accuracy degrades over time as it predicts further from the training period, the monthly MAE was calculated over the 17-month test period from July 2023 to November 2024. This analysis is crucial since the model is trained on historical data it may become less accurate as production conditions change. The monthly MAE ranged from 4.72 to 8.27 days with no clear upward or downward trend over time. The highest error occurred in December 2023 (MAE 8.27 days), which may have been caused by some external production disruptions not captured in the ERP data. Over the 17-month test window, no degradation trend was observed, which suggests that the relationship between product configuration and lead time is relatively stable within this period. Longer stability cannot be stated from this result and periodic retraining of the model remains important if the model is deployed for production planning, especially when a new product families or high-volume variants are introduced. Figure 15 shows the monthly MAE over the test period.

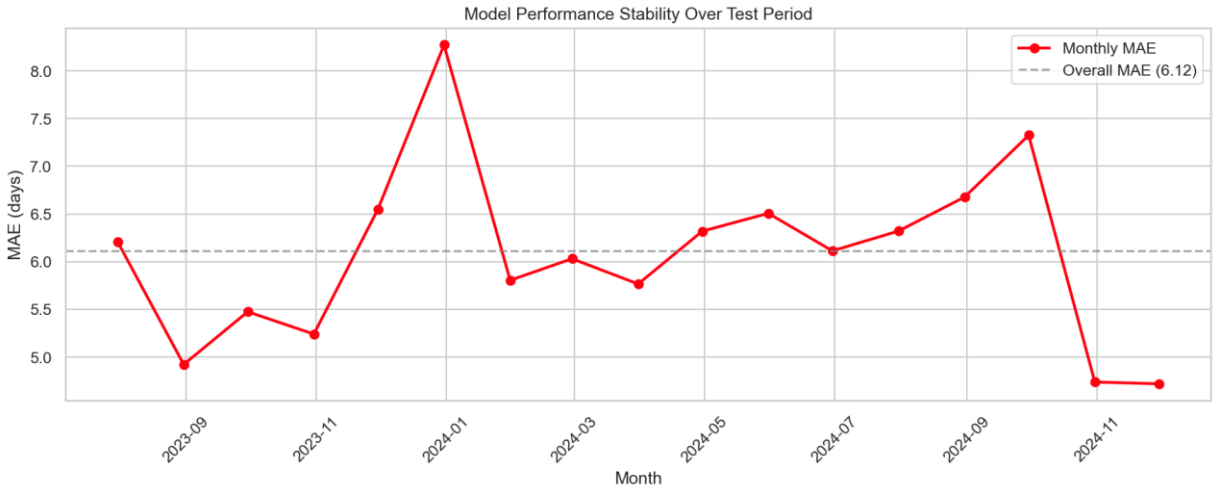

Figure 15. Monthly MAE over the test period.

6.2 Key Factors Influencing Lead Time

This section uses SHAP analysis to answer RQ3 by identifying which product configuration features have the strongest influence on the model's lead time predictions. The analysis covers global feature importance, the relative contribution of different feature categories and the directional effect of individual variant codes.

6.2.1 Global Feature Importance

SHAP analysis reveals which features the model relies on most when predicting lead times. Each feature receives a SHAP value for every prediction, measured in business days. A positive SHAP values mean the feature pushes the prediction upwards, while a negative value pushes it downward. The mean absolute SHAP value for a feature represents the average impact on predictions across all orders, regardless of the direction. A higher value means the feature has stronger influence on the model's output.

The most important feature is FrameSize with a mean absolute SHAP values of 1.98 days. This means that on average, knowing the frame size of a motor shifts the predicted lead

time by about 2 days up or down compared to the base prediction. Larger frame sizes correspond to physically larger motors that require more material, longer machining and more complex assembly which naturally increases the production duration.

The second most important feature is Variant 711, with a mean absolute SHAP value of 1.38 days. This is about 70% of the mean absolute SHAP value of FrameSize, placing Variant 711 clearly as the second influential feature and showing that certain product customizations have a strong and consistent effect on production duration. The third most important feature is VariantCount with a mean absolute SHAP value of 0.83 days, which represents the total number of variant codes on an order. This suggests that the overall level of customization, regardless of which specific variants are selected, is itself a predictor for lead time. Table 8 below shows the top 15 most important features.

Table 8. Top 15 features by mean absolute SHAP value.

Feature	Mean absolute SHAP value (days)
FrameSize	1.98
Variant 711	1.38
VariantCount	0.83
Variant 999	0.68
MotorFamily M3AA	0.31
Variant 755	0.31
Variant 710	0.30
PolePairs	0.20
MotorFamily M3EH	0.19
Variant 107	0.19
MotorFamily M3BP	0.17
Variant 179	0.16
Variant 466	0.14
MotorFamily M3LP	0.14
MotorFamily M3GP	0.13

Figure 16 shows the SHAP summary plot, which provides more detail than the bar chart by displaying each individual observation as a dot. In this plot, features are listed on the vertical axis ordered by importance and the horizontal axis shows the SHAP value. Each dot represents one order from the sample, and its color indicates the feature values, red for high values and blue for low values. For FrameSize, the red dots (large frame sizes) are spread to the right meaning longer lead times, while the blue dots (small frame sizes) are on the left meaning shorter lead times. For binary features such as Variant 711, red dots (variant present) appear far to the right, showing that the presence of this variant adds substantially to the lead time. The SHAP summary plot for MotorFamily_M3FT shows red dots far to the right which is consistent with the high lead times observed for this newer motor family.

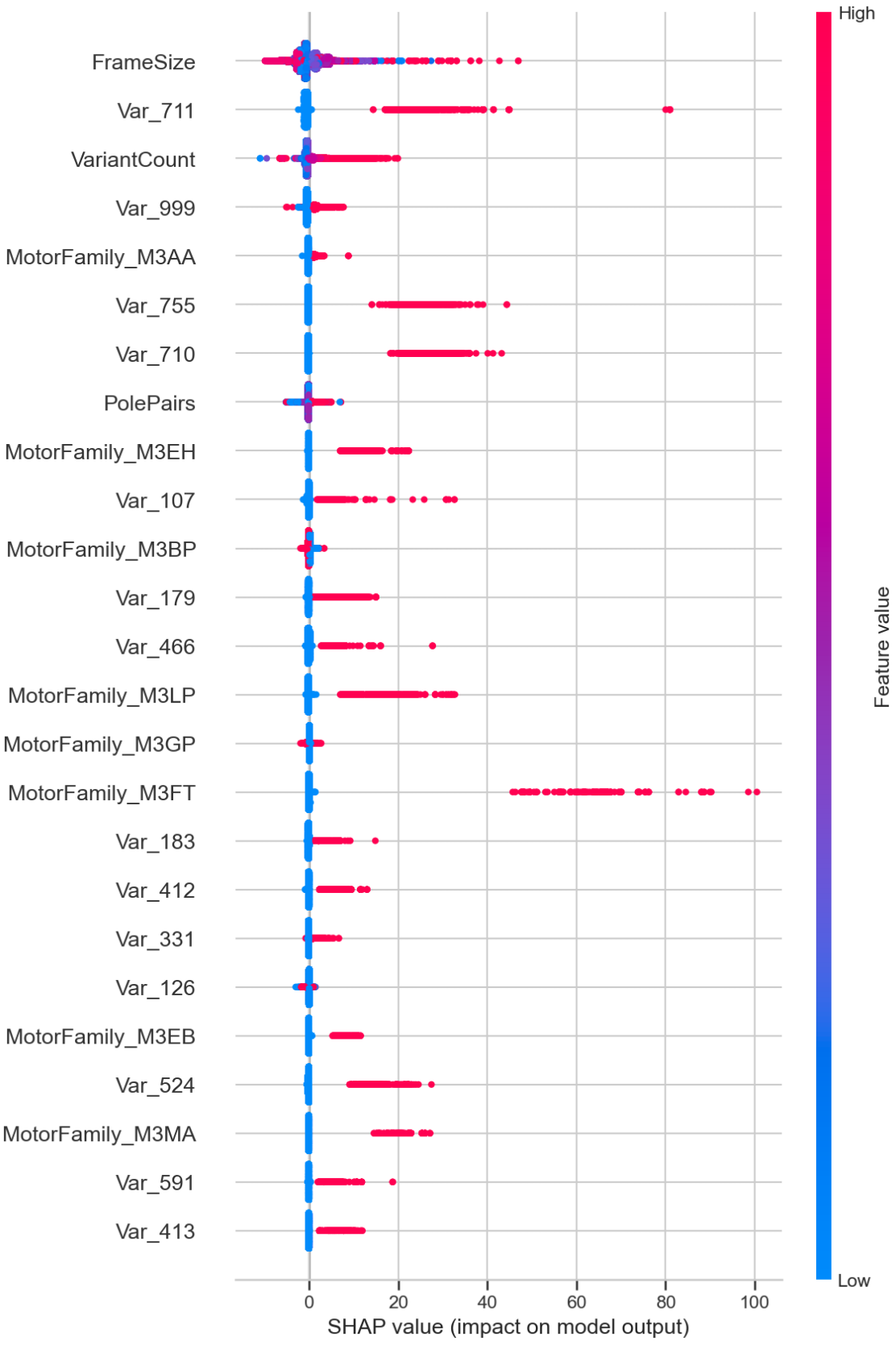

Figure 16. SHAP summary plot (top 25 features).

6.2.2 Feature Importance by Category

When the 487 individual features are grouped into three major categories, the relative importance reveals a pattern. The individual variant codes account for 55.3% of the total SHAP importance, motor properties (motor family, frame size and pole pairs) that are derived from the basic description account for 36.4% and the variant count accounts for 8.2%.

This distribution shows that more than half of the model's predictive power comes from the specific product customizations applied to each order, rather than from the base motor properties. In other words, knowing which variant codes are on an order is more important for predicting lead time than knowing the motor family and frame size. This finding supports the premise of this study that product configuration data, particularly variant codes contain valuable information for predicting production lead times. It also suggests that any lead time estimation system that relies only on the base motor type without considering specific customization option is leaving a significant part of the information unused.

6.2.3 Feature Effects on Lead Time

The SHAP analysis reveals which specific variant codes according to the model increase or decrease lead time when they are present on an order. This is computed by averaging the SHAP values only for orders where each variant is present. This answers to the question when this variant is included in an order, how much does it increase or decrease the expected lead time. The effect is measured relative to the model's base prediction which is 25.8 days. Table 9 and Table 10 show the variant codes with the strongest effect on lead time in days when present. These effects were computed on the full training set of 159,904 order lines. The count column reports how many training orders contain each

variant and variants with low counts should be interpreted with caution since the mean is based on only few observations.

Table 9. Variant codes that increase lead time when present.

Variant	Effect when present	Count (present)
713	+31.6 days	35
710	+27.3 days	875
711	+23.6 days	4,720
755	+23.5 days	1,013
719	+23.0 days	48
712	+20.8 days	44
524	+14.5 days	351
385	+12.4 days	25
517	+7.6 days	11
590	+7.1 days	85
413	+6.1 days	869
877	+5.3 days	467
412	+5.1 days	1,511
015	+4.8 days	337
466	+4.8 days	2,196
764	+4.2 days	154
182	+4.0 days	469
035	+3.9 days	107
591	+3.9 days	1,628
843	+3.9 days	706

Table 10. Variant codes that decrease lead time when present.

Variant	Effect when present	Count (present)
602	-3.5 days	1
729	-3.2 days	277
782	-2.2 days	1,099
011	-1.8 days	793
251	-1.1 days	833
010	-1.0 days	432
543	-0.6 days	746
762	-0.6 days	2,516
096	-0.5 days	5,543
643	-0.5 days	47

The results show a clear group among the variants that increase lead time. Six variant codes (713, 710, 711, 755, 719 and 712) all add well over 20 days to the predicted lead time when present. These variants are all related to different types of special coating that require processing at an external paint shop. The motors must be transported to the external facility, processed and returned before production can continue, which explains the consistent three-to-four-week delay. Among this group variant codes 711 is by far the most common in the group. It appears in 4,720 training set orders, while other variants in the group are rare, with 712, 713 and 719 being the rarest with under 50 appearances each in the training set.

Below the special coating variants, there are a couple of variants that add a relatively high number of days to the predicted lead time, forming a middle tier. Variant 524 adds 14.5 days to the predicted lead time and appears in 351 orders, and variant 385 adds 12.4 days but appears only in 25 orders. The remaining variants in the top 20 add roughly 4-6 days while still being moderately important for the analysis due to their appearance in many of the orders. For example, variant 466 is present in 2,196 orders, variant 591 in 1,628 orders and variant 412 in 1,511 orders. Their frequency in the orders means that

together they can have a substantial effect on the predicted lead time even though their per-order effect is much smaller than the special coating variants. The variant codes that decrease lead time show a different nature. The negative effects range only between -1 and -3 days, which indicates that the variant codes tend to add lead time rather than decrease it. This observation is consistent with the nature of make-to-order manufacturing where additional specifications require additional processing steps meaning a longer time in production.

The same conditional averaging approach was applied to motor families to show their effects on the predicted lead time. Table 11 shows the effect of each motor family when present on an order.

Table 11. Effect of each motor family when present.

Motor Family	Effect when present	Count (present)
M3ET	+67.7 days	27
M3FT	+63.9 days	147
M3LK	+51.9 days	13
M3MA	+20.7 days	278
M3LL	+19.6 days	13
M3LP	+15.1 days	687
M3EH	+9.6 days	1,488
M3EB	+7.4 days	767
M3BL	+6.1 days	781
M3AL	+5.2 days	320
M3RP	+3.8 days	341
M3BJ	+3.5 days	164
M3JM	+2.6 days	15
M3BC	+1.7 days	660
M3AA	+1.3 days	20,505
M3AR	+0.5 days	1,242
M3JP	+0.0 days	9,171
M3KP	-0.1 days	25,206
M3BP	-0.2 days	78,169
M3GP	-0.6 days	18,943
M3HP	-1.2 days	736

The motor family analysis reveals a strong relationship between production volume and lead time effect. The two families with the largest effects on the lead time (M3ET and M3FT) are the same motor families that showed the highest prediction errors in Section 6.1.5. Their large SHAP effects confirm that the model has learned to associate these families with a longer production lead time, though the high MAE indicates that the actual lead times for these families varies a lot and are hard to predict. The model treats

high-volume motor families such as M3BP, M3JP, M3KP and M3GP as the baseline because they present 82% of the training set. Their minimal impact on the lead time predictions show that the production processes of these motor families are most standardized.

Figure 17 shows SHAP dependence plots for two features FrameSize and VariantCount. These plots show how the SHAP value of these features depend on the feature's value, with each dot representing one order in the training set. The FrameSize dependence plot shows that the smallest frame sizes (71-90) have SHAP values mostly positioned between 0 and -5 days, while the largest frame sizes (450-500) have strong positive effect ranging from +10 to over +40 days. However, the relationship is not only linear and the SHAP value is not straight growing with the frame size. Frame sizes in the middle range (132-315) show wide vertical spread, with SHAP values ranging from approximately -10 to +15 days. This indicates that the frame size alone does not fully determine its effect on the lead time, other features on the order also influence how much a frame size adds to the lead time. The VariantCount dependence plot shows a clear growing linear relationship between the number of variant codes to the SHAP value. Orders with 0-5 variant codes tend to have effect between -3 to 5 days meaning they are near the average prediction. As the number of variant codes increases, the spread of SHAP values also widens, which indicates that the type of variants present on the orders matters also, certain combinations of variants produce a larger effect on lead times than other combinations.

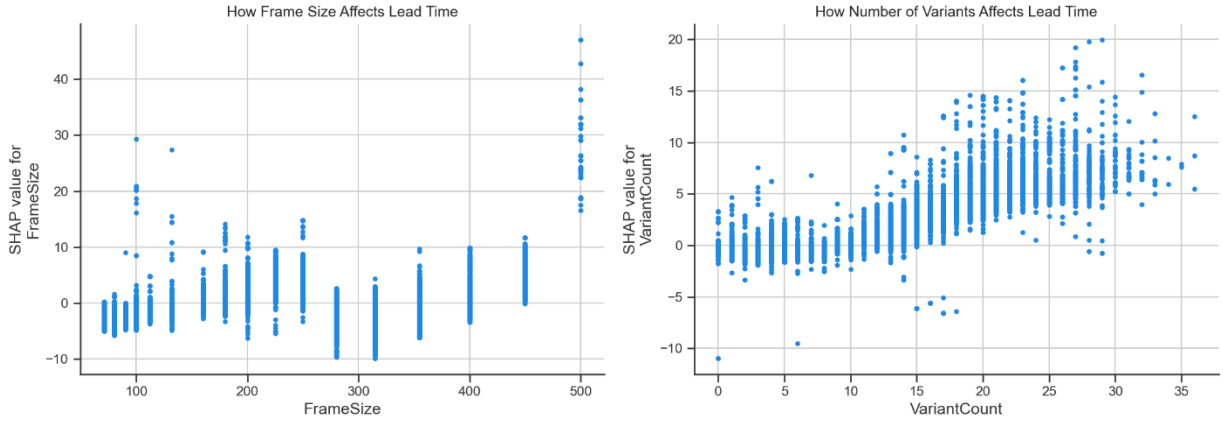

Figure 17. SHAP dependence plots for FrameSize and VariantCount.

6.3 Case Examples

To illustrate how the developed model makes individual predictions, three production orders were randomly selected from different lead time ranges in the test set, one with a short actual lead time under 20 days, one with a medium lead time between 20-40 days and one with a long lead time over 40 days. There were generated SHAP waterfall plots for each example the shows how each feature affects the prediction from the average of 25.8 days to get to the model's prediction.

The first example is an order with an actual lead time of 19.0 days. The model predicted 22.2 days, overestimating the lead time by 3.2 days. The waterfall plot (Figure 18) shows that nearly every feature pushed the prediction down from the base value (25.8 days). The only feature pulling the prediction up among the top 15 features was variant 398 (+0.74 days). FrameSize of 112 contributed -0.65 days, which reflects the results shown in the dependence plot of frame size. Smaller frame sizes have shorter lead times. The absence of variants 711 (-0.64 days) and 999 (-0.44 days) decreased the lead time prediction and the VariantCount of 1 decreased it even further by -0.43 days. The remaining visible variants from the top 15 decreased the lead time prediction but with smaller negative effects between -0.11 and -0.26 days and the 473 other features together contributed with -0.83 days. This a small motor with low customization level which the model

correctly recognized as a below-average order. The 3.2-day error shows that the model does not fully capture how fast such small and simple order proceeds in production.

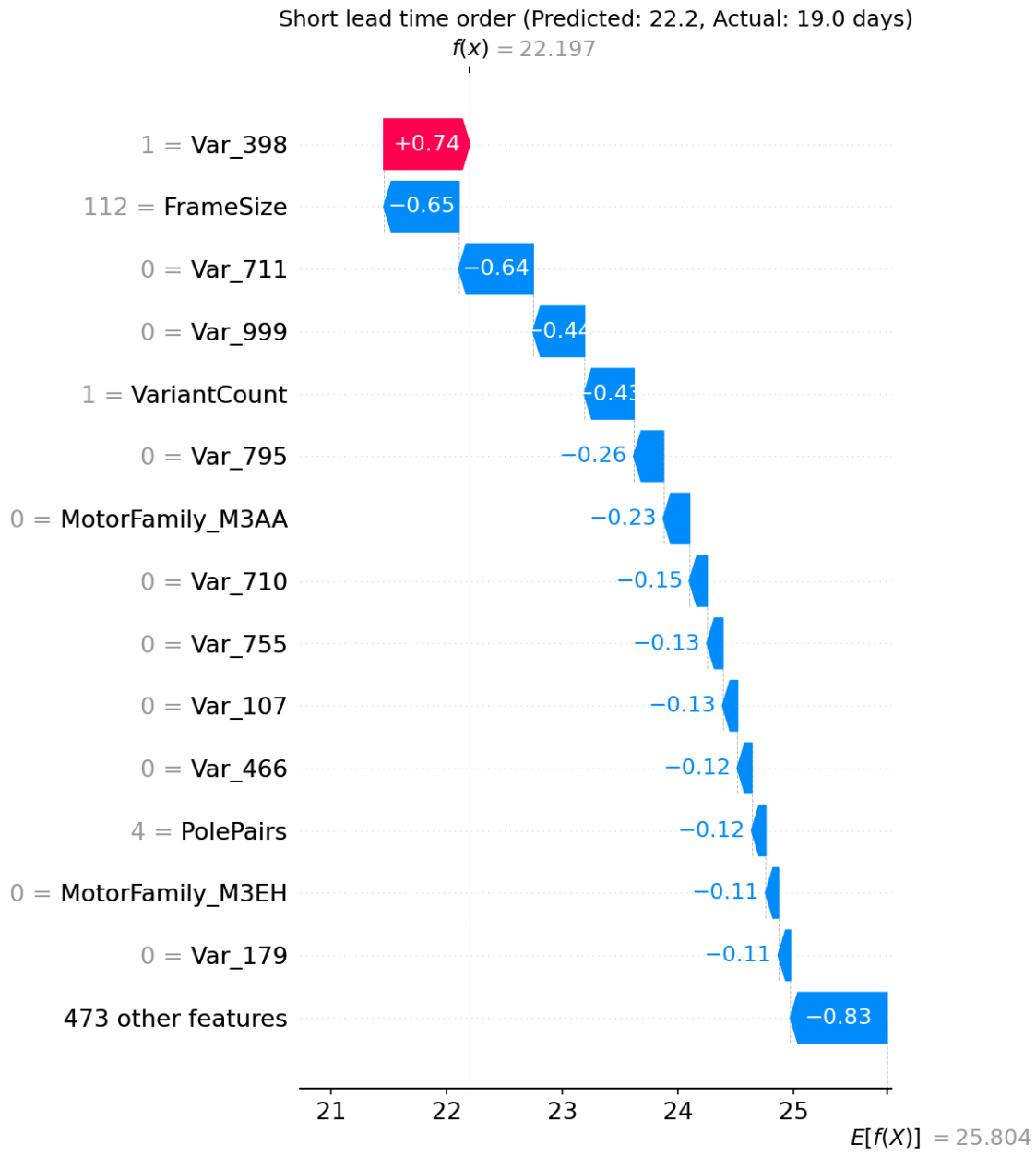

Figure 18. SHAP waterfall plot for a short lead time order.

The second example is an order with an actual lead time of 30.0 days. The model predicted 28.9 days, underestimating just by 1.1 days. This is an M3BP motor with frame size 315 and high VariantCount of 21. The waterfall plot (Figure 19) shows that the

VariantCount was the dominant contributor at 4.71 days, which is consistent with the finding in Section 6.2.3 that highly customized orders take longer to produce. Variant 129 added +1.86 days and variant 999 added +1.76 days. However, FrameSize 315 pulled the prediction down by -1.56 days which is consistent with dependence plot in Section 6.2.3, showing the effect frame sizes going down to around 315 frame sizes. The absence of variant 711 reduced the prediction again with the reduction of -0.78 days and the common motor family M3BP reduced the prediction by -0.41 days. Several other variants reduced the prediction such as 754 by -0.39 days and 126 by -0.29 days, but also the absence of variant 755 related to special coating reduced the time by -0.24 days. The prediction landed very close to the actual lead time with an error of 1.1 days.

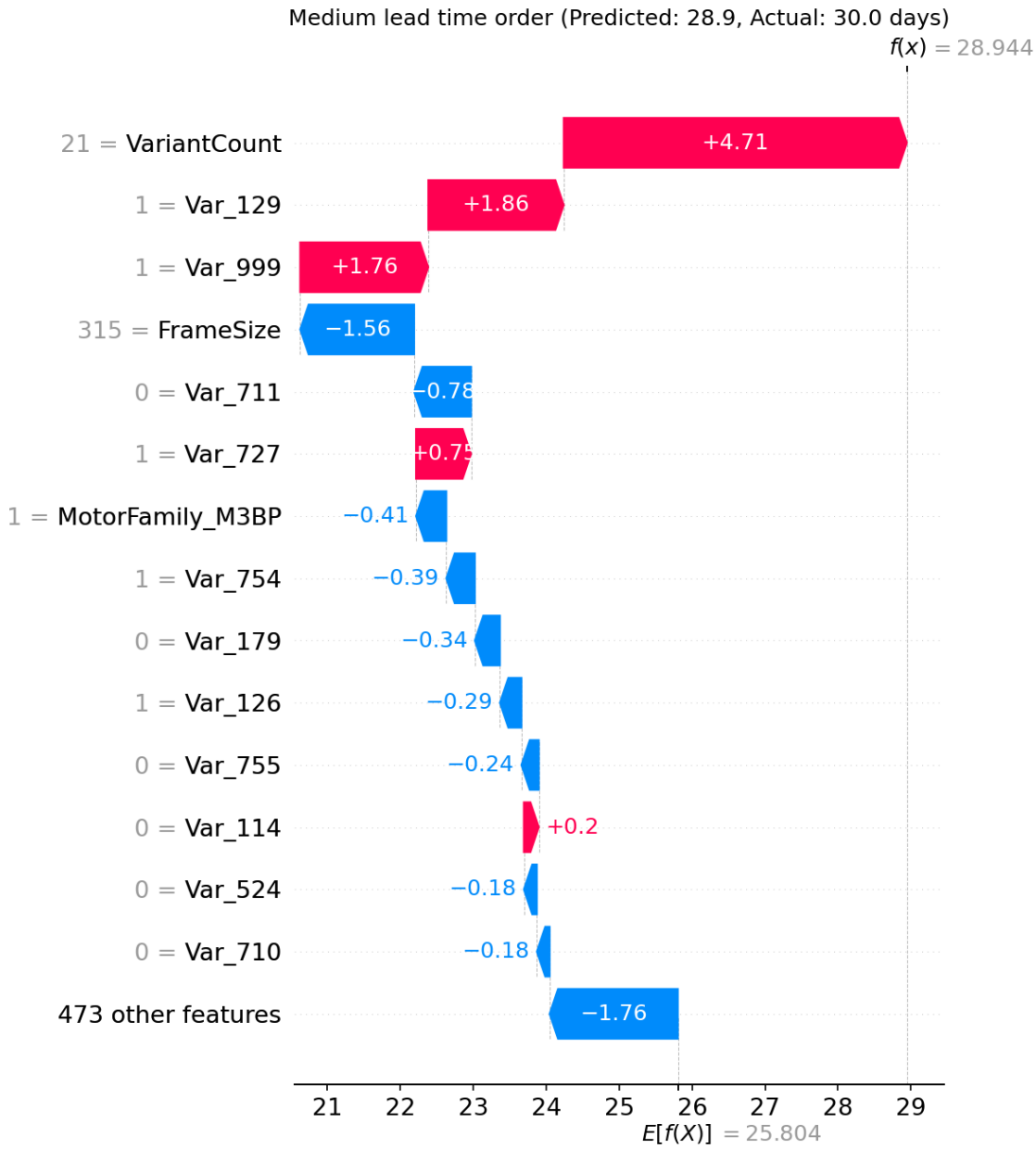

Figure 19. SHAP waterfall plot for a medium lead time order.

The third example from the test set is an order with an actual lead time of 91.0 days. The model predicted 66.6 days, underestimating by 24.4 days. The waterfall plot (Figure 20) is visually showing one single feature affecting the most to the prediction, variant 710 which contributes +32.76 days. This confirms the earlier results that the presence of the special coating variants increases the prediction remarkably. FrameSize feature has the value of 225 which also adds the lead time by +5.35 days, while less dominant variants

such as 178, 746 and 194 add together approximately +3 days. The remaining features had little to no effect on the prediction but in total the model predicted over 40 days longer lead time than the average. This shows that even though there is a large prediction error of 24.4 days, the model correctly identified this as an order that would take far longer than the average in production which is valuable information for the production planning.

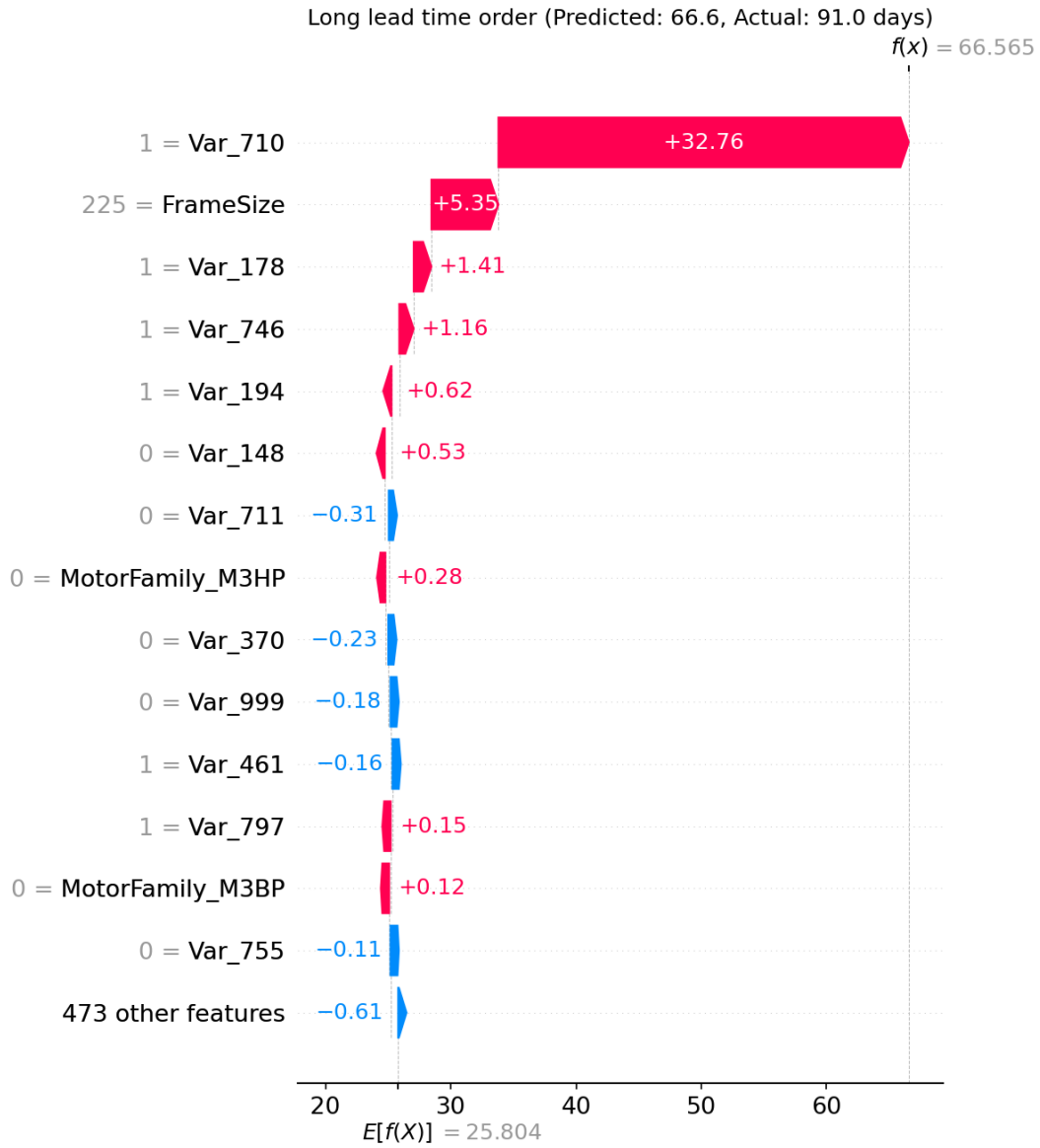

Figure 20. SHAP waterfall plot for a long lead time order.

Together, these three examples show how the model uses the product configuration data to adjust the predictions in different situations. For simple orders with minimal customization, features pull the prediction below the average when highly customized orders have much higher predictions but also a bigger prediction error.

7 Discussion

This study investigated the potential of machine learning models to predict production lead times in the case-company's manufacturing context using their historical ERP data. The research addressed three main questions: how effectively machine learning can use ERP data for predicting production lead time, which algorithm performs best, and which product characteristics most significantly influence lead time predictions. This chapter interprets the key findings, discusses their practical and theoretical implications, acknowledges limitations of the study, and suggests directions for possible future research.

7.1 Reflection on the Results

The first research question asked how effectively machine learning models can predict production lead times using ERP data. The results show that all three machine learning models outperformed the ERP system's planned lead times. The best model, tuned XGBoost, achieved a MAE of 6.12 days compared to the ERP system's 8.12 days, representing a 24.7% reduction in average prediction error. The cross-validation MAE of 6.24 days with a 95% confidence interval of [4.55, 7.93] confirms that this improvement is stable across different time periods. The model's R^2 of 0.31 indicates that product configuration data alone explains about one third of the variance in the actual lead times. This is a meaningful result considering that the model uses only static product attributes and does not have access to dynamic production conditions such as machine availability, workload or supply chain status.

The fact that the naive baseline with MAE of 7.89 days performed comparably to the ERP system is itself a notably finding. It suggests that the current ERP planning approach captures very little of the variation in lead times and operates essentially at the level of global average. The machine learning models on the other hand leaned to differentiate between orders based on their product configurations, achieving better predictions without having any information about the current state of production.

The second research question asked which machine learning algorithm provides the most accurate predictions. With library default parameters, LigthGBM achieved the lowest MAE of 6.29 days, followed closely by XGBoost with MAE of 6.32 days and Random Forest at last with MAE of 6.84 days. After hyperparameter tuning all three algorithms with Optuna the ranking changed. Random Forest showed the largest improvement from 6.84 to 6.23 days, which was a reduction of 8.9%, but XGBoost became the best performing model with an MAE of 6.12 days. LightGBM's performance remained unchanged at 6.29 days. This finding shows that algorithm rankings can change substantially after hyperparameter tuning. The close performance of all three algorithms in this study suggests that the choice of an algorithm matters less than having a well-designed feature set and proper temporal validation.

The third research question asked which product configurations have the most significant impact on production lead time. The SHAP analysis revealed that individual variant codes account for 55.3% of the total feature importance, motor properties such as motor family, frame size and pole pairs account for 36.4% and the variant count accounts for 8.2%. This means that the specific product customizations applied to each order are more informative for predicting lead time than the base motor type. The most important single feature was FrameSize with a mean absolute SHAP value of 1.98 days, followed by Variant 711 with 1.38 days, which corresponds to a special coating performed at an external paint shop. Six coating-related variants (713, 710, 711, 755, 719 and 712) each added over 20 days to the predicted lead time when present on order, which makes them the strongest individual drivers for extended production duration. Additionally, the motor family analysis revealed that newer motor families such as M3ET with +67.7 days and M3FT +63.9 days add substantial lead time when present. These findings have a practical value for the case company, they can quantify better the lead time impact of specific customization options which could've been previously hidden.

Comparing these results to prior studies, the 24.7 % improvement over the existing planning system is in line with what has been reported in previous studies. Rokoss et al. (2024) used workload and capacity features that are not available in the present study, which may explain why their models could capture a larger share of variance. Burggräf et al. (2020) found that material data was used in only 5% of lead time prediction studies they reviewed. While their review showed that physical material properties are underutilized, this study demonstrates that a broader category of product-specific data and specifically customization options such as the variant codes in this case also contains a lot of predictive information that has not been utilized in prior research that much. This suggests that the use of product data is not limited only to the physical properties of materials and more generally covers motor assembly information and different production phases.

The model's accuracy varied a lot across different product types. For the four most common motor families such as the M3BP, M3JP, M3GP and M3KP which account for 91% of the order lines used in this study, the MAE was between 4.95 and 5.68 days. For the newer motor families such as the M3FT and M3ET, the MAE exceeded over 34 days which shows the high variability between product types well. The monthly MAE over the 17-month test period shows no degradation trend which suggests that the learned patterns remain valid within this window. The test window is too short to draw conclusions about longer stability of the model.

7.2 Practical Implications

The findings of this study offer several practical contributions to the case company. First, the tuned XGBoost model can provide more accurate lead time estimates than the current ERP planning system for most production orders. For the previously mentioned most common motor families, the model's average error of approximately 5 days is a significant improvement over the ERP system's average error of 8 days. Implementing this model as a tool alongside the existing planning system could help production planning make more realistic delivery commitments to customers.

Second, understanding which product configurations most significantly affect production lead times allows for more precise resource allocation and capacity planning. The identification of six external special coating variants (713, 710, 711, 755, 719 and 712) as the strongest individual drivers of lead time, each adding over 20 days when present, gives the case company specific customization options to investigate. Among these variants, variant 713 showed the largest effect of +31.6 days but appeared in only 35 training set orders, while variants 710, 711 and 755 are more common and each add over 23 days, which makes these most reliable and practical targets for further investigation. The company could examine whether the external paint shop process can be optimized or whether alternative solutions or whether delivery date estimates for orders with these variants should be adjusted in the current planning system. Similarly, the finding that VariantCount is itself predictor of lead time suggests that highly customized orders require often more production time, which could also be factored into planning. These insights can start investigations in the case company to understand why certain product configurations add to the lead time and some do not. This could potentially lead to improvements in the production processes.

Third, the explainability provided by the SHAP analysis helps to understand how the machine learning model makes predictions. Rather than having the machine learning model operating as a black box, the SHAP analysis provides transparent explanations for the model's predictions through waterfall plots that show which features push the prediction up or down for each individual prediction. As demonstrated with three randomly selected orders from different lead time ranges from the test set in Section 6.3 of this study. This helps production planning understand the model better and allows them to override the model's prediction when they have better knowledge of the current situation on the shop floor, such as machine breakdowns or problems in the supply chain.

Fourth and final practical implication for the case company is the developed prediction utility function as part of the created artifact, which allows the model to be applied to new orders from an Excel file without requiring user to run full machine learning pipeline.

This functionality acts as an easily accessible tool for production planning, that helps to make better estimates and more reliable delivery commitments.

7.3 Theoretical Contributions

From a theoretical perspective, this study makes several contributions to the literature on machine learning in manufacturing. First, it demonstrates that product configuration data derived from an ERP system holds a lot of predictive power for lead time prediction. This finding is related to the gap identified by Burggräf et al. (2020), who found that material data such as physical dimensions of products appeared in only 5% of lead time prediction studies when reviewing previous literature on the topic. While variant codes are not identical to the material data category in their framework, both represent product-specific information that has been underutilized for lead time prediction. The results of this study show that product-related data sources beyond order data and system status contain substantial information for predicting lead times.

Second, the study contributes to the growing literature on explainable AI in manufacturing contexts. By demonstrating how SHAP analysis makes complex machine learning models interpretable for production planning decisions, the research shows that predictive performance and interpretability do not need to be mutually exclusive. With hundreds of one-hot encoded features, standard SHAP output provides limited insight on its own. This study shows that grouping features into more meaningful categories and analyzing their effects produces more structured and informative interpretation of model's behavior. This approach can be applied in other manufacturing contexts where high dimensional data makes raw feature importance and ranking difficult to interpret.

Third, the finding that all three algorithms changed their relative ranking after hyperparameter tuning contributes to the ongoing discussion about algorithm comparison methodology. This result suggests that studies comparing algorithms only with their default

settings may reach different conclusions than those that tune all algorithms, which has implications for how algorithm comparisons are designed for future research.
Fourth, the study adds to the literature on ERP data utilization for advance analytics. This research demonstrates that substantial value can be extracted from existing data inside a manufacturing company through appropriate machine learning techniques, even without requiring additional data sources. This finding is particularly relevant for small and medium-sized manufacturing companies who may lack resources for extensive data infrastructure investments.

7.4 Limitations and Future Research

This study has several limitations that should be considered when assessing the results. First limitation is that the study is based on data from a single manufacturing company producing electric motors. While the dataset is large (approximately 200,000 order lines), the findings may not generalize to other manufacturing contexts or product types. The focus on specific product configuration options and the processes related to those are unique to the case company and the relative importance of different features would likely be different in another manufacturing setting.

Second, the model uses only product configuration features derived from the ERP system and does not include dynamic production conditions such as current machine availability and capacity, workforce availability, capacity of different production processes or supply chain status. The R^2 of 0.31 indicates that a substantial share of the variance in actual lead times remains unexplained and much of this unexplained variance is likely caused by these dynamic factors not included in the dataset. Including real-time shop floor data as demonstrated in studies by Gyulai et al. (2018) and Rokoss et al. (2024), could remarkably improve the prediction accuracy.

Third, the model identifies correlations between product configurations and lead times but does not establish causal relationships. For example, the model may learn that

certain variant codes are associated with longer lead times without distinguishing whether the variant itself causes the delay or whether it is correlated with other factors that do cause the delay. This is an important difference for process improvement, acting on the correlation findings from this study without understanding the underlying mechanisms or processes that cause the delay could lead to ineffective improvements. To turn these findings into actual process improvements, the production processes behind these variants need to be examined further.

Fourth limitation is the training data which covers the period from January 2019 to July 2023, which includes also the COVID-19 pandemic. Production conditions during this period were likely atypical with supply chain disruptions and changes in demand that may have influenced the patterns learned by the model. Although the stable performance over the 17-month test period suggests this effect is limited, the possibility of this cannot be fully ruled out.

Fifth, the data quality issues observed in the ERP system, such as negative planned lead time down to -224 business days indicate that the ERP dates are not always reliable. This affects the ERP baseline comparison and suggests that other fields in the dataset could contain similar errors and inconsistences that were not detected during the data preprocessing.

There is also a fundamental limitation of any model trained with historical data, it cannot predict accurately for newly introduced products or variants. In practice the model should be retrained periodically as new product families are introduced and their production processes stabilize or when new high-volume variants emerge that could affect the model's prediction accuracy.

Several directions for future research emerge from these limitations. Integrating dynamic shop floor data such as machine availability, queue lengths and workforce availability could address the unexplained variance and improve the accuracy of the model.

Expanding the study to multiple manufacturing companies or products would test the generalizability of the approach. Developing a system that updates the model continuously as new production data becomes available could address the limitations with newer product families and variants and adapt to changing production conditions. Although the study by Lingitz et al. (2018) found that tree-based ensembles perform best across multiple different machine learning algorithms, expanding the amount of comparable methods could reveal whether different algorithms capture patterns that the tree-based methods miss. Finally combining the predictive model with optimization methods could enable not just better lead time estimation but also active production scheduling that minimizes lead times. In summary, these new research directions point towards more agile prediction systems that combine historical order data with real-time production conditions.

8 Conclusion

This study developed and evaluated a machine learning artifact for predicting production lead times in a make-to-order electric motor manufacturing company. Using approximately 200,000 historical production orders from the company's ERP system, three tree-based ensemble algorithms were trained and compared against the company's existing planning system.

The results show that the hyperparameter tuned XGBoost model reduced the average prediction error by 24.7% compared to the ERP system's planned lead times with an MAE of 6.12 days versus the ERP system's 8.12 days. For the four most common motor families, which account for 91% of production orders, the model achieved a MAE of approximately 5 days. The model's accuracy remained stable over a 17-month test period with no degradation trend, suggesting that the learned relationships between product configuration and lead times are stable over time.

XGBoost outperformed LightGBM and Random Forest after hyperparameter tuning, although LightGBM had the lowest error with default parameters. Random Forest benefited the most from tuning with 8.9% improvement in the mean average error but still could not surpass XGBoost. This highlights the importance of hyperparameter tuning when comparing machine learning algorithms as the relative ranking changed after optimization.

The SHAP analysis revealed that product customization data, specifically in this case, variant codes account for 55.3% of the model's predictive power. Several special coating-related variants that require processing at an external paint shop were identified as the strongest individual drivers of extended lead time, each adding over 20 days when present on an order. These findings demonstrate that product configuration data, which has been underutilized in prior lead time prediction research, contains substantial information for predicting production duration.

The study contributes to practice by providing the case company with a validated prediction tool and actionable insights into which product configurations most significantly affect production duration. The SHAP-based explanations enable production planning to understand and trust the model's predictions, supporting the possible adoption in manufacturing environment where transparency is important. The study contributes to theory by demonstrating how explainable machine learning can be integrated into production planning and by extending the limited use of product-related data in lead time prediction research.

The main limitation is that the model explains approximately 31% of variance in the actual lead time, with the remaining variance likely driven by dynamic production conditions not available in the input data from the ERP system currently. Future research could improve prediction accuracy by integrating real-time shop floor data such as machine availability and queue lengths and by expanding the approach to other manufacturing contexts to test its generalizability. Overall, the study demonstrates how historical ERP data, when combined with appropriate machine learning methods and interpretability tools, can support accurate and transparent lead time planning in make-to-order manufacturing.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1907.10902

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213. https://doi.org/https://doi.org/10.1016/j.ins.2011.12.028

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Burggräf, P., Wagner, J., Koke, B., & Steinberg, F. (2020). Approaches for the Prediction of Lead Times in an Engineer to Order Environment—A Systematic Review. *IEEE Access*, *8*, 142434–142445. https://doi.org/10.1109/ACCESS.2020.3010050

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, 785–794. https://doi.org/10.1145/2939672.2939785

Fahle, S., Prinz, C., & Kuhlenkötter, B. (2020). Systematic review on machine learning (ML) methods for manufacturing processes – Identifying artificial intelligence (AI)

methods for field application. *53rd CIRP Conference on Manufacturing Systems 2020*, *93*, 413–418. https://doi.org/10.1016/j.procir.2020.04.109

Gyulai, D., Pfeiffer, A., Bergmann, J., & Gallina, V. (2018). Online lead time prediction supporting situation-aware production control. *6th CIRP Global Web Conference – Envisaging the Future Manufacturing, Design, Technologies and Systems in Innovation Era (CIRPe 2018)*, *78*, 190–195. https://doi.org/10.1016/j.procir.2018.09.071

Hevner, A., R, A., March, S., T, S., Park, Park, J., Ram, & Sudha. (2004). Design Science in Information Systems Research. *Management Information Systems Quarterly*, *28*, 75.

Ioannou, G., & Dimitriou, S. (2012). Lead time estimation in MRP/ERP for make-to-order manufacturing systems. *Compassionate Operations*, *139*(2), 551–563. https://doi.org/10.1016/j.ijpe.2012.05.029

Jodlbauer, H., & Strasser, S. (2019). Capacity-driven production planning. *Computers in Industry*, *113*, 103126. https://doi.org/10.1016/j.compind.2019.103126

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*.

Kingsman, B. G., Tatsiopoulos, I. P., & Hendry, L. C. (1989). A structural methodology for managing manufacturing lead times in make-to-order companies. *European Journal of Operational Research*, *40*(2), 196–209. https://doi.org/10.1016/0377-2217(89)90330-5

Köber, J., & Heinecke, G. (2012). Hybrid Production Strategy Between Make-to-Order and Make-to-Stock – A Case Study at a Manufacturer of Agricultural Machinery with Volatile and Seasonal Demand. *45th CIRP Conference on Manufacturing Systems 2012*, *3*, 453–458. https://doi.org/10.1016/j.procir.2012.07.078

Lingitz, L., Gallina, V., Ansari, F., Gyulai, D., Pfeiffer, A., Sihn, W., & Monostori, L. (2018). Lead time prediction using machine learning algorithms: A case study by a semi-conductor manufacturer. *51st CIRP Conference on Manufacturing Systems*, *72*, 1051–1056. https://doi.org/10.1016/j.procir.2018.03.148

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S.-I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, *2*(1), 56–67. https://doi.org/10.1038/s42256-019-0138-9

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

Öztürk, A., Kayalıgil, S., & Özdemirel, N. E. (2006). Manufacturing lead time estimation using data mining. *European Journal of Operational Research*, *173*(2), 683–700. https://doi.org/10.1016/j.ejor.2005.03.015

Pargent, F., Pfisterer, F., Thomas, J., & Bischl, B. (2022). Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. *Computational Statistics*, *37*(5), 2671–2692. https://doi.org/10.1007/s00180-022-01207-6

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst.*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Pfeiffer, A., Gyulai, D., Kádár, B., & Monostori, L. (2016). Manufacturing Lead Time Estimation with the Combination of Simulation and Statistical Learning Methods. *Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems*, *41*, 75–80. https://doi.org/10.1016/j.procir.2015.12.018

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, 1135–1144. https://doi.org/10.1145/2939672.2939778

Rokoss, A., Syberg, M., Tomidei, L., Hülsing, C., Deuse, J., & Schmidt, M. (2024). Case study on delivery time determination using a machine learning approach in small batch production companies. *Journal of Intelligent Manufacturing*, *35*(8), 3937–3958. https://doi.org/10.1007/s10845-023-02290-2

Santoso, L. & Priyadi. (2024). Comparative Study of Feature Engineering Techniques for Predictive Data Analytics. *Journal of Technology Informatics and Engineering*, *3*(2), 417–435. https://doi.org/10.51903/jtie.v3i2.225

Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, *81*, 84–90. https://doi.org/10.1016/j.inffus.2021.11.011

Usuga Cadavid, J. P., Lamouri, S., Grabot, B., Pellerin, R., & Fortin, A. (2020). Machine learning applied in production planning and control: A state-of-the-art in the era

of industry 4.0. *Journal of Intelligent Manufacturing*, *31*(6), 1531–1558. https://doi.org/10.1007/s10845-019-01531-7

Valdez-Valenzuela, E., Kuri, A., & Gomez Adorno, H. (2021). *Measuring the Effect of Categorical Encoders in Machine Learning Tasks Using Synthetic Data* (pp. 92–107). https://doi.org/10.1007/978-3-030-89817-5_7

Verdonck, T., Baesens, B., Óskarsdóttir, M., & Vanden Broucke, S. (2024). Special issue on feature engineering editorial. *Machine Learning*, *113*(7), 3917–3928. https://doi.org/10.1007/s10994-021-06042-2

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, *30*(1), 79–82. JSTOR.

Zheng, A., & Casari, A. (2018). *Feature engineering for machine learning: Principles and techniques for data scientists*. O'Reilly Media, Inc.