Predicting processing times in high mix low volume job shops

B.Sc. Anna Müller
Center for Applied Data Science (CfADS), Bielefeld University of Applied Sciences, Gütersloh, Germany
anna.mueller@hsbi.de

M.Sc. Felix Grumbach
Center for Applied Data Science (CfADS), Bielefeld University of Applied Sciences, Gütersloh, Germany

DOI: http://dx.doi.org/10.25673/103491

Abstract
Production planning is essential for any manufacturing company, especially when complex and varied processes must be considered. Accurate processing times play a critical role for scheduling production runs and allocating resources effectively. In practice, the respective master data from the ERP system is often used for this purpose. However, maintaining the master data is challenging, especially with large amounts of data in flexible environments. In this context, incorrect or outdated data quickly lead to significant planning inaccuracies. This paper presents a study that uses machine learning (ML) models to accurately predict the processing times of single operations of future production runs based on historical production runs. Various ML algorithms were trained and evaluated on a real-world dataset. In comparison to the master data the root mean squared error could be reduced by 23% using ML. Thus, these estimated times can be used for optimizing future schedules and incorporating such an ML model in the production planning process eliminates the need for master data.

1. Introduction
This section provides a brief overview of the motivation behind this study, summarizes the key findings and limitations of related research in the field and points out the contribution of the proposed work.

1.1. Motivation
Production planning is essential for any manufacturing company, especially when complex and varied processes must be considered [1]. Accurate processing times are crucial to effectively schedule production runs and allocate resources [2]. However, processing times can vary significantly in reality, which makes achieving optimal schedules difficult. Currently, simple estimates, such as the average time, are commonly used, but they can be imprecise and result in inefficient and suboptimal schedules, particularly for high mix low volume manufacturers with many products, resources and flexible processes [2,3]. Addressing the problem of varying processing times has been a subject of extensive research, with two approaches emerging: using simple probability distributions to describe the variation of processing times or using ML to predict the actual processing times [1]. In recent years, ML has become increasingly important in various fields, with rapid developments in algorithms and model architectures, decreasing costs of sensors and computing hardware, and an explosion in data volume fueling interest in ML applications in production [4]. ML enables computer programs to perform complex tasks, such as prediction, diagnosis, planning, and recognition, by learning from historical data [3].

1.2. Related Work
A considerable amount of literature has been published on production optimization. A great deal of research focuses either on developing effective scheduling algorithms or on the estimation of time-related KPIs in the production environment. The former presupposes given operation times without questioning their validity, such as [5–7]. However, valid processing times are needed for robust results of the optimization procedures. The latter focuses on estimating the cycle time instead of the duration of individual operations [3], such as [8–11]. There is a relatively small body of literature that is concerned with the prediction of processing times. Sobaszek et al. [12] provide a statistical approach to estimate the distribution of the processing times based on historical data. Both Mucientes et al. [13] and Ringsquandl et al. [2] utilized regression models to predict the actual processing times. A key shortcoming of these

studies is their reliance on expert knowledge. Since expert knowledge can be scarce and lacks scalability, there is a need for scalable and automated approaches to predict processing times in a production field. It should be noted that the data used in [2] was generated from simulation and thus, needs further validation with real-world data. Yamashiro and Nonaka [1] tested several ML models on a real-world dataset. Product ID, number of products, materials and material parameters were used as input features for the utilized ML models. They also suggest analyzing the importance of the input features for more scalability and generalizability.

1.3. Contribution

This paper presents a study that assesses several ML models, classic linear and non-linear ones, to accurately predict the processing times of single operations of future production runs based on historical production runs. As shown in Figure 1, the processing times in our dataset do not have a symmetrical distribution.

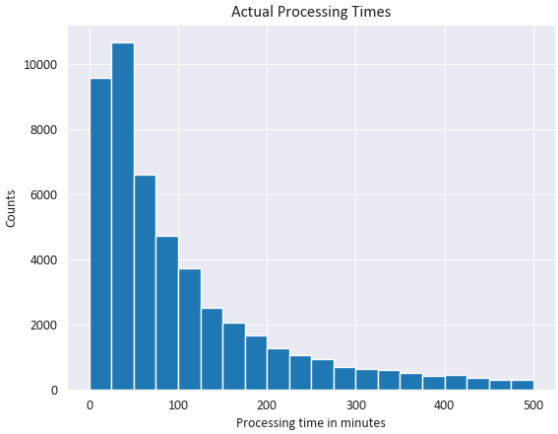

Figure 1: Histogram of all actual processing times of the real-world dataset

Since linear models such as LR require a normally distributed target variable [14], we opted to predict the deviation of the actual duration of an operation and the planned duration. The distribution of the deviation is approximately normal, as shown in Figure 2.

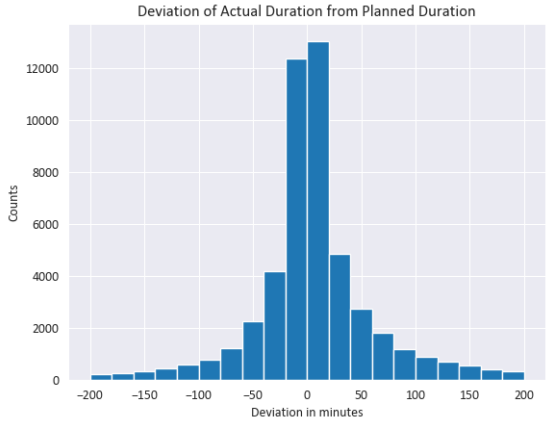

Figure 2: Histogram of all deviations of the actual duration from the planned duration

The task of predicting the deviation is treated as a regression problem. Based on several input features, which contain different information of a scheduled operation, the deviation is estimated as a continuous target variable.

Moreover, the trained ML models were utilized to determine features, that highly influence the deviation. The insights gained from the feature analysis for practitioners were also evaluated.

This article is organized as follows: In Section 2, the data preprocessing, the utilized ML algorithms and the results of the hyperparameter tuning are described. The results of the regression models, the influence of individual features, as well as strength and weaknesses of the model are analyzed in Section 3. Finally, in Section 4 the results are summarized, and potential future work is discussed.

2. Methods or experimental part

In this section, the dataset, the utilized ML models and the employed experimental design are described.

2.1. Dataset

The dataset to train and evaluate the ML models is provided by one of our industrial research partners. It is a medium-sized supplier company in the mechanical engineering sector, characterized by discontinuous high mix low volume production. The dataset consists of booking data from historical work processes from the flexible shop floor with machine tools, assembly stations, various qualified employees, external workbenches, and complex bill of materials. It contains about 52,000 operations from the last seven years and includes the following features: Actual start and end time, planned processing time, product, quantity, machine, assigned employee and job due date. Personal data has been anonymized so that no conclusions can be drawn about specific employees.

2.2. Data pre-processing and feature engineering

To ensure the quality of the data, we systematically cleaned it, removing errors and outliers. Due to the great variety of products, we used k-Means clustering to group the products based on weight, number of subparts, frequency of production and value. The results of the clustering were verified with a domain expert of the company. To improve the accuracy of the prediction, several features were engineered. The timestamps were used to create time-related categorical features such as day of week and

month as additional input features and the deviation of processing time, as target feature. In addition, features to describe operation predecessors, employee utilization and machine breakdowns were modeled. The data was then standardized, and a univariate feature selection was performed to select the best 20 features. A more detailed analysis of the most influential features and their impact on the production planning processes can be found in Section 3.2.

2.3. Machine Learning

Several ML algorithms, specifically linear regression (LR), support vector regression (SVR), decision tree (DT) regressor, random forest (RF) regressor and multi-layer perceptron (MLP), were employed using scikit-learn [15] to predict the deviation of the processing time of an operation. To evaluate the performance of these models, we split the dataset into a training and a test set. For each algorithm, except for linear regression, which has no hyperparameters, hyperparameter tuning was performed on the training set using grid search with 5-fold cross-validation.

2.3.1. Linear Regression

Linear regression is a statistical regression method. Let us assume that Y denotes the target variable and x1, …, xn are the input variables. The objective is to find the parameters β0, …, βn for a prediction Y' with

$$Y' = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n \quad (1)$$

so that the sum of squared errors of prediction and target variable is minimized [16]. Since this algorithm has no hyperparameters, no hyperparameter tuning was performed.

2.3.2. Support Vector Regression

SVM is a ML algorithm initially developed for classification problems. The objective is to find the hyperplane that separates two classes of data points with the maximal margin. The width of the margin is determined by the smallest distance to one of the data points. SVMs can also be adapted for regression problems. The idea of margin is still present, but instead of classifying data points into two separate classes, the data points are now supposed within a margin around the regression line. Slack variables are associated with outliers that lie outside the margin, and the algorithm tries to minimize the sum of these slack variables. The support vectors, namely margin vectors and outlier vectors, determine the regression curve [17]. During the hyperparameter tuning, two essential hyperparameters were examined: the kernel function and the regularization parameter C. The kernel defines a mapping of the input data to a higher-dimensional space, which enables non-linear regression [17]. The regularization parameter controls the weights of the slack variables.

Table 1: Hyperparameter Tuning SVR. The best parameter value is marked

	Assessed parameter values
Kernel	Linear, **Polynomial**, Radial basis function, Sigmoid
C	**0.1**, 1, 10

2.3.3. Decision Tree Regressor

DTs are a commonly used ML algorithm for complex relationships between input variables and output variables. The algorithm works by constructing a tree-like model, where at each internal node in the tree a binary test is applied to one of the input variables to split the samples into smaller subsets [18]. Each split aims to minimize the residual sum of squares between the average value and actual values of the target variable of all data points belonging to the corresponding subset [17]. This process is repeated until a termination criterion is met. All data points are summarized in a leaf node, where the actual prediction is made. The predicted value is typically the average target value of all data points belonging to that leaf [18]. Since overfitting is quite common [17], we tested several stopping criteria: the maximal depth of the tree, the minimum number of samples required in the child node after splitting and the minimum number of samples required for a leave.

Table 2: Hyperparameter Tuning Decision Tree Regressor. The best parameter value is marked

	Assessed parameter values
Maximal depth	4, **6**, 8, 10, 12, 14, 16, 18, 20
Minimum samples split	10, 20, **30**
Minimum samples leave	**5**, 10, 15

2.3.4. Random Forest Regressor

A RF is an ensemble learning technique that consists of several DTs. Each tree is trained separately, and each node is split using a randomly selected subset of features. The final prediction is the average prediction of all trees [17]. In general, this technique is more robust to noisy data than a single DT [19].
The hyperparameter tuning covers the same hyperparameters as the hyperparameter tuning for the decision tree regressor and additionally includes the number of decision trees used for the ensemble learner.

Table 3: Hyperparameter Tuning Random Forest Regressor. The best parameter value is marked

	Assessed parameter values
Maximal depth	4, 6, 8, 10, 12, 14, 16, **18**, 20
Minimum samples split	10, **20**, 30
Minimum samples leave	**5**, 10, 15
Number of estimators	100, **1000**

2.3.5. Multi-Layer Perceptron

MLPs also known as neural networks, consists of several neurons, which are arranged in layers. The inputs of the MLP are weighted and propagated to the first layer of neurons, where each neuron passes the weighted sum of inputs through an activation function, typically the ReLu function. The outputs are passed as inputs to the next layer [17]. In the last layer, the so-called output layer, all inputs are combined for the actual prediction. This structure enables learning complex non-linear relationships between input and output variables. During the hyperparameter tuning, different structures and sizes of the hidden layers were tested, along with different initial learning rates and learning rate changes.

Table 4: Hyperparameter Tuning MLP. The best parameter value is marked

	Assessed parameter values
Hidden layer size	(300,200,100), (200,100,50), (50,50,50), **(100,100,100)**, (50,50), (100,100), (200,200), (200,100,50)
Initial learning rate	**0.0001**, 0.001, 0.01
Learning rate	**Constant**, adaptive, invscaling

3. Results and Discussion

The performance of the ML models and their applicability for the regression task and the importance of the input features are analyzed in this section. Additionally, the tree-based models are examined in more detail.

3.1. Regression analysis

All models were evaluated on a separate test dataset using the rooted mean squared error (RMSE) as evaluation metric. RMSE is a commonly used metric for evaluating the predictive errors of regression models. A smaller RMSE value indicates a higher model performance. The planned durations from the ERP system's master data were used as benchmarks. RMSE values for the deviation of the master data and all ML models are shown in Figure 3.

Figure 3: RMSE for predicted and actual deviation on the test set

The results show that all models exhibit a similar performance, which is significantly better compared to the master data. It indicates a great potential for the general use of ML models to predict the deviation of the processing time. In practice, the choice of model seems to be a secondary concern. However, the Random Forest Regressor generated the most accurate predictions on our test dataset. In comparison to the master data, the root mean squared error (RMSE) could be decreased by 23 % using the predictions of the RF. Furthermore, the standard deviation of the error could be decreased by 22%. It implies, that the model is capable of providing a more precise representation of the data and a more accurate prediction of the deviation.

3.2. Feature analysis

Further analysis of the influence of the input features on the deviation was conducted. Since the RF showed the best results, this model was utilized for the analysis. Scikit's RF has an in-build feature called feature importance. Every time a feature is chosen in a splitting node, it reduces the impurity. The normalized total reduction is equal to the feature importance [15]. The table below shows the five most important features according to the RF's feature importance.

Table 5: Five most frequently chosen features chosen by RF

Feature	Feature importance
Planned duration	**0.747**
Weekly workload	0.105
Lot size	0.0702
Operation thirteen	0.0202
Machine twenty-five	0.0170

All the features, except weekly workload, have a negative correlation with the predicted deviation. This means, that high values of the features lead to larger negative deviations. The planned duration exhibits the strongest negative correlation with the deviation with a correlation coefficient of -0.836. Thus, it can be concluded that in this practical use case especially long processing times were overestimated in the planning stage. Interestingly, only the feature weekly workload shows a positive correlation with the predicted deviation. In this case, the weekly workload describes the amount of work in hours that is already scheduled for the employee that is executing this operation. The findings indicate the necessity of conducting a comprehensive examination by practitioners to explore the underlying causes of the observed correlation. One possible approach to mitigate the effect could involve the implementation of an equitable workload distribution strategy among employees.

3.3. Detailed model analysis

Tree-based models are a popular choice in ML due to their high level of transparency. Thus, the decisions of the DT model and the strength and weaknesses of the RF model are examined in more detail.

3.3.1. Decision analysis

The DT model, which demonstrated a comparable performance on the test set, is highly interpretable and transparent. The tree structure enables easy visualizations of the decision-making process, as the paths from the root node to the leaf nodes represent a sequence of decisions that lead to the final prediction. Figure 4 depicts such a graphical representation of the first three tiers of the DT model. The representation was automatically generated by utilizing the open-source graph visualization software graphviz [20]. The representation reaffirms the importance of the planned duration as a feature in predicting the deviation. In the first two tiers the DT splits the samples into groups purely based on their planned duration. The largest group, comprising 95.2% of all samples, also contains the samples with the lowest planned duration. The samples in the other groups have a higher planned duration, indicating operations with a higher lot size or more complex operations. Interestingly, in the third tier the DT not only uses the planned duration but also the binary feature Operation thirteen for splitting. As can be seen, the tree is very complex and cannot be fully examined in this paper. Nonetheless, it is evident that the tree serves as a critical launching point for domain experts' discussions.

3.3.2. Performance analysis

Returning to the RF, we conducted a thorough analysis of the strength and limitations of the incorporated model. On the one hand, it demonstrated an outstanding performance for operations with a moderate lot size and planned duration. A moderate planned duration might indicate a decreased probability of interruptions, which unpredictably influence the processing time. Moreover, a moderate Lot size of products suggests that the products are usually not unique

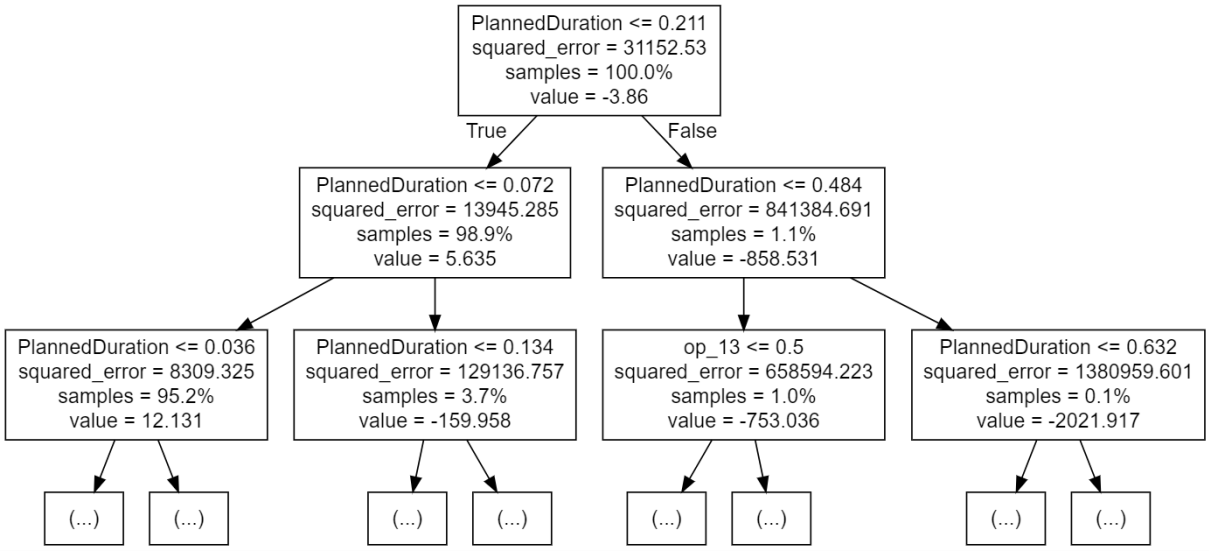

Figure 4: The first three tiers of the DT model, automatically generated using graphviz

items. As such, unique products are often characterized by their complexity and distinctive nature, making them difficult to compare with other known products.
On the other hand, the RF did not perform well in predicting outcomes for products belonging to cluster 20. This cluster is known for containing products that require significant amount of manual work and consists of multiple subparts. Thus, these products are exceptionally complex, and the model might need additional features to improve the accuracy of the predictions. We assume that this is a challenging problem, as even experts such as technicians, designers and production planners apparently struggle with estimating the time for complex custom parts. Nevertheless, we consider this to be an interesting and relevant research need for the future, as custom manufacturing and small lot sizes play an important role in the German industrial landscape [21].

4. Limitations and Conclusion

This study presents a novel approach for estimating processing times using ML. To the best of our knowledge, this is the first study in the field of processing time prediction that utilizes clustering to deal with the high variety of products. Thus, this approach is in practice particularly interesting for high mix low volume manufacturers. Additionally, this study is the only one so far that uses self-generated time-related features and information about machines and workers for the regression model in addition to product-related features. The results have shown that all models performed significantly better than the master data manually maintained in the ERP system. The RF performed slightly better than the other models, reducing the RMSE by 23% in comparison to the master data. Additionally, the standard deviation of the error could be reduced by 22%, which highlights the robustness of the proposed approach.
Since the regression model does not require expert knowledge, it enables a highly automated prediction process that can be used for optimizing future schedules online. Incorporating such an ML model in the production planning process leads to more efficient and effective use of resources and eliminates the extensive master data maintenance. Moreover, the additional feature analysis offers valuable insights for practitioners. Although these insights might not be directly transferable to other companies, the process is again completely automated and therefore transferable.
However, the study also highlights the importance of appropriate training data. The provided data is somewhat limited and noisy, underscoring the need for more comprehensive and reliable data in future studies to enhance the accuracy of the ML models [3]. Furthermore, with more available data, it would be possible to directly predict the processing time instead of the deviation. It would also be interesting to evaluate the influence on production schedules, that incorporate ML to predict the processing time.

5. References

[1] Yamashiro, H., Nonaka, H. (2021). Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem. Operations Research Perspectives 8, 100196.

[2] Ringsquandl, M., Lamparter, S., Lepratti, R. (2015). Estimating Processing Times within Context-aware Manufacturing Systems. IFAC-PapersOnLine 48/3, 2009–2014.

[3] Kang, Z., Catal, C., Tekinerdogan, B. (2020). Machine learning applications in production lines: A systematic literature review. Computers & Industrial Engineering 149, 106773.

[4] Krauß, J., Hülsmann, T., Leyendecker, L., Schmitt, R. H. (2023). Application Areas, Use Cases, and Data Sets for Machine Learning and Artificial Intelligence in Production. In: Production at the Leading Edge of Technology. Liewald, M., Verl, A., Bauernhansl, T., Möhring, H.-C. (Hrsg.). Springer International Publishing, Cham, 504–513.

[5] Al-Behadili, M., Ouelhadj, D., Jones, D. (2020). Multi-objective biased randomised iterated greedy for robust permutation flow shop scheduling problem under disturbances. Journal of the Operational Research Society 71/11, 1847–1859.

[6] Gonzalez-Neira, E. M., Montoya-Torres, J. R., Jimenez, J.-F. (2021). A Multicriteria Simheuristic Approach for Solving a Stochastic Permutation Flow Shop Scheduling Problem. Algorithms 14/7, 210.

[7] Wang, H., Sarker, B. R., Li, J., Li, J. (2021). Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning. International Journal of Production Research 59/19, 5867–5883.

[8] Biazon de Oliveira, M., Zucchi, G., Lippi, M., Cordeiro, D., Da Rosa Silva, N., Iori, M. (2021). Lead Time Forecasting with Machine Learning Techniques for a Pharmaceutical Supply Chain. In: Proceedings of the 23rd International Conference on Enterprise Information Systems. SCITEPRESS - Science and Technology Publications, 634–641.

[9] Gyulai, D., Pfeiffer, A., Nick, G., Gallina, V., Sihn, W., Monostori, L. (2018). Lead time prediction in a flow-shop environment with

analytical and machine learning approaches. IFAC-PapersOnLine 51/11, 1029–1034.
[10] Bender, J., Ovtcharova, J. (2021). Prototyping Machine-Learning-Supported Lead Time Prediction Using AutoML. Procedia Computer Science 180, 649–655.
[11] Alnahhal, M., Ahrens, D., Salah, B. (2021). Dynamic Lead-Time Forecasting Using Machine Learning in a Make-to-Order Supply Chain. Applied Sciences 11/21, 10105.
[12] Sobaszek, Ł., Gola, A., Kozłowski, E. (2019). Prediction of variable technological operation times in production jobs scheduling. IFAC-PapersOnLine 52/13, 1301–1306.
[13] Mucientes, M., Vidal, J. C., Bugarin, A., Lama, M. (2008). Processing times estimation in a manufacturing industry through genetic programming. In: 2008 3rd International Workshop on Genetic and Evolving Systems. IEEE, 95–100.
[14] Poole, M. A., O'Farrell, P. N. (1971). The Assumptions of the Linear Regression Model. Transactions of the Institute of British Geographers 52, 145.
[15] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É. (2012). Scikit-learn: Machine Learning in Python.
[16] Draper, N. R., Smith, H. (1998). Applied Regression Analysis. Wiley.
[17] Braga-Neto, U. (2020). Fundamentals of Pattern Recognition and Machine Learning. Springer International Publishing, Cham.
[18] Gordon, A. D., Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). Classification and Regression Trees. Biometrics 40/3, 874.
[19] Breiman, L. (2001). Random Forests. Machine Learning 45/1, 5–32.
[20] Gansner, E. R., North, S. C. (2000). An open graph visualization system and its applications to software engineering. Softw: Pract. Exper. 30/11, 1203–1233.
[21] Schmieder, M. (2018). Hidden Champions Benchmarking. In: Fallstudienkompendium Hidden Champions. Büchler, J.-P. (Hrsg.). Springer Fachmedien Wiesbaden, Wiesbaden, 197–222.