Management and Production Engineering Review
Volume 17 • Number 1 • March 2026 • pp. 1–11
DOI: 10.24425/mper.2026.158632

The Application of Supervised Machine Learning Algorithms for Improving the Accuracy of Manufacturing Operation Duration Prediction

Jadwiga Krupnik-Worek[1], Sebastian Skoczypiec[2]

[1] *Cracow University of Technology, CUT Doctoral School, Faculty of Mechanical Engineering, Department of Production Engineering and Automation, Poland*
[2] *Cracow University of Technology, Faculty of Mechanical Engineering, Department of Production Engineering and Automation, Poland*

Received: 03 November 2025
Accepted: 08 February 2026

Abstract
The study presents an approach to predicting the completion time of production operations using supervised machine learning techniques. The analysis was conducted on a database extracted from ERP and MES systems, comprising over 150,000 records containing technological and production completion times, operation numbers, and textual descriptions of operations. The dataset preprocessing involved cleaning, feature encoding, and text vectorisation using the TF-IDF method to represent semantic patterns within operation descriptions. Regression models, including Linear Regression, Random Forest, and XGBoost, were trained and evaluated using Google Colab. Model performance was assessed using standard evaluation metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). Experimental results showed that ensemble-based methods achieved the highest predictive accuracy, outperforming the baseline model based solely on technological completion time. In addition, the study examines the sensitivity of selected models to hyperparameter settings and analyses the impact of alternative categorical feature encoding methods on prediction accuracy. The proposed approach enables more accurate estimation of production durations and supports data-driven decision-making in manufacturing environments.

Keywords
Machine learning, Production planning, Supervised learning.

Introduction

The execution time of technological operations in the production process depends on numerous technical, organisational, and human factors. These include the technical condition of the machinery, the operator's experience and availability, and the complexity and accuracy required for the processing. As indicated in the literature (Sobaszek et al., 2017), the actual execution times often differ significantly from those planned by technologists, creating a significant source of uncertainty in production process planning. Such discrepancies can lead to outdated production schedules, impaired material flow and reduced resource efficiency. Consequently, they can lead to increased costs and shorter delivery times. Therefore, it is particularly important to identify the causes and nature of variation in technological operation times and to develop methods for effectively predicting them.

In recent years, artificial intelligence and machine learning methods have become increasingly popular for this purpose. These methods enable the modelling of complex, nonlinear relationships between process parameters, operational characteristics, and execution time. Among these approaches, regression algorithms can be used to develop predictive decision support systems that enable more accurate production planning and dynamic schedule adjustments based on actual production conditions.

Accurately predicting the duration of technological operations in manufacturing requires robust methods that estimate future outcomes from historical data. Machine learning methods can be used for this purpose, and these are divided into four main algorithms, as shown in Figure 1 (Miller et al., 2024).

***Corresponding author**: Jadwiga Krupnik-Worek – Cracow University of Technology, CUT Doctoral School, Faculty of Mechanical Engineering, Department of Production Engineering and Automation, Poland, e-mail: jadwiga.krupnik-worek@doktorant.pk.edu.pl*

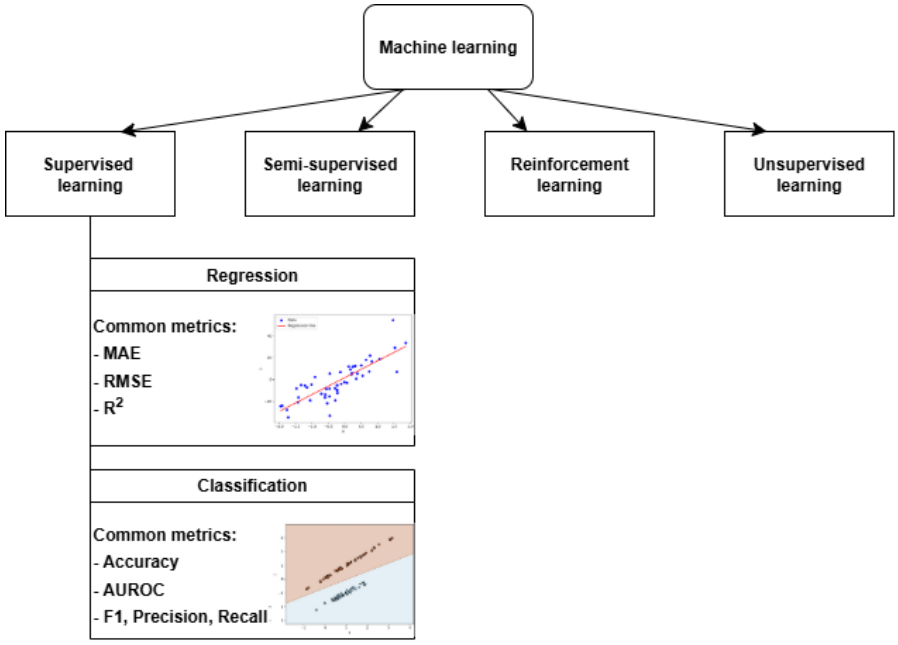

Fig. 1. Division of machine learning into four categories. This paper focuses on two supervised learning categories: classification and regression

The goal of the presented study is to develop a methodology and present a case study of the application of supervised machine learning techniques to forecast operation durations, enabling more accurate scheduling and optimised resource utilisation.

Literature review

Predictive Modelling Techniques

Predictive modelling is an advanced analytics field that uses historical and current data to forecast future outcomes. By employing statistical and computational algorithms, it identifies underlying patterns which are then applied to predict values for new, previously unknown cases. This multidisciplinary approach integrates techniques from statistics, machine learning, database management, and optimization (Kumar & Garg, 2018; Lamba & Madhusudhan, 2022).

The primary objective of these models is to develop methods that accurately predict target values based on specific input features. To achieve this, the field frequently relies on supervised learning, a process where the model learns to map input variables to output values using labelled training data (Lamba & Madhusudhan, 2022).

Predictive models can be broadly divided into the following categories (Kumar & Garg, 2018):

- *Classification models*, which predict class membership (e.g. the type of operation)
- *Regression models*, which predict numerical values (e.g. operation execution time)

This work focuses on regression models for predicting the time required for various production processes.

Linear regression is one of the simplest and most used regression techniques. The aim is to model the relationship between a dependent variable (Y, execution time) and one or more independent variables (X, input features). The model assumes that changes in the target variable are proportional to changes in the features. This allows the impact of each feature on the predicted value of Y to be estimated (Kumar & Garg, 2018). Linear regression involves fitting coefficients to minimise the sum of squared differences between the actual and predicted values (mean squared error). It can be used to examine various properties of the data, such as linearity and homoscedasticity, as well as the significance of the predictors (Grandhi, 2019). In the presented experiment, linear regression served as a simple benchmark and interpretive tool. It enabled the basic relationships between features and execution time to be assessed. When the relationships are more complex and cannot be captured well by linear models, nonlinear methods such as decision trees and ensemble techniques are used.

Decision trees represent a form of classification model, yet they are equally applicable in the context of regression. They divide data into subsets based on feature

values, forming a tree-like structure comprising decision nodes and leaves containing predictions. Although simple trees are intuitive, they often exhibit high variance and can be prone to overfitting. To counteract these limitations, ensemble learning methods are employed to aggregate multiple weak models and produce a stronger predictor (Kumar & Garg, 2018; Tober, 2020).

Random Forest is an ensemble method that creates multiple decision trees from random subsets of data and features. The results from the individual trees are then averaged (in the case of regression), reducing variance and improving predictive stability. Random Forest is particularly useful for complex and nonlinear data (Tober, 2020). While Random Forest performs well with nonlinearity, more advanced ensemble techniques, such as XGBoost, also iteratively correct prediction errors, thereby increasing prediction precision.

Extreme Gradient Boosting (XGBoost) is a machine learning algorithm that belongs to the *Gradient Boosting Machines* (GBM) family. This model creates an ensemble of decision trees, with each subsequent tree adjusting the prediction errors of those that came before. This enables XGBoost to identify complex, nonlinear relationships between features and the target variable (Chen & Guestrin, 2016; Grandhi, 2019). XGBoost was originally designed as a classification and regression tool rather than a time-series forecasting system. However, research has shown that, with appropriate data processing and preparation, it can be used effectively for time-series forecasting, such as predicting company turnover (Wójcik, 2018).

In summary, linear regression can serve as a simple benchmarking and interpretation tool. Meanwhile, Random Forest and XGBoost allow us to model complex, nonlinear relationships between features and the target variable.

Evaluation Metrics

Evaluation metrics are a key element of both regression analysis and predictive machine learning models. These mathematical frameworks quantify the alignment between predicted and observed values, enabling standardised comparisons of performance across different models. The closer the actual (r) and predicted (p) values are, the better the predictive model's performance (Grandhi, 2019; Plevris et al., 2022).

In machine learning, these measures assess how effectively a trained model can forecast outcomes for an unseen test set. Comparing predictions with ground-truth observations is essential for evaluating a model's validity, precision, and ability to generalise to new data (Miller et al., 2024; Plevris et al., 2022). Selecting the right evaluation metrics allows us to measure the accuracy of these predictions and make an objective assessment of which model best reflects the relationships in the data.

In regression analysis, several well-established metrics are commonly used to evaluate model performance. The most widely applied of these are the mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R^2) (Lamba & Madhusudhan, 2022; Miller et al., 2024; Plevris et al., 2022).

The mean absolute error (MAE) is a measure of the average magnitude of prediction errors. It provides an intuitive indication of the average deviation between predicted and actual values (1). A lower MAE indicates a smaller average difference between the predicted and observed values (Miller et al., 2024; Plevris et al., 2022).

$$\mathrm{MAE} = \sum_{i=1}^{n} \frac{|y_i - x_i|}{n} \tag{1}$$

Given:
- x_i – predicted value i
- y_i – true value i
- n – number of data points

Another commonly used measure is the Root Mean Squared Error (RMSE), which penalises larger errors more strongly due to squaring. It is expressed as follows (2) (Miller et al., 2024; Plevris et al., 2022).

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}} \tag{2}$$

Given:
- x_i – predicted value i
- y_i – true value i
- n – number of data points

Although both MAE and RMSE measure average prediction error, RMSE is more sensitive to outliers. This makes it particularly useful in situations where large deviations are undesirable (Drzewiecki, 2017; Miller et al., 2024; Plevris et al., 2022; Rainio et al., 2024).

The coefficient of determination (R^2) measures the proportion of the variance in the dependent variable that the model (3) explains. Higher R^2 values (closer to 1) indicate that the model explains a greater proportion of the variability in the data, thus providing a better fit (Ampuła, 2017; Lamba & Madhusudhan, 2022; Miller et al., 2024; Plevris et al., 2022).

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - x_i)^2}{\sum_{i=1}^{n} (y_i - y_{\mathrm{mean}})^2} \tag{3}$$

Given:

- x_i – predicted value i
- y_i – true value i
- y_{mean} – mean of true values
- n – number of data points

These metrics are widely recognised and consistently applied in academic research and industrial practice. Examples include studies on predictive modelling in manufacturing, transportation delay forecasting and bioinformatics (Drzewiecki, 2017; Miller et al., 2024; Wójcik, 2018).

The next sections present assumptions and the results of an experiment comparing the accuracy of selected machine learning models in predicting production lead times. This enables evaluation of their usefulness in production planning and resource scheduling. For this study, we chose MAE, RMSE, and R^2 metrics as clear and widely accepted indicators of prediction accuracy and model fit.

Experiments

Data preprocessing and feature engineering

The experiments were conducted using real production data extracted from the ERP and MES systems, comprising 157 312 records collected between January and November 2020. Each record contains information about a technological operation, including its description, number, planned duration according to the technological plan, and actual completion time recorded in production. As some operations were recorded multiple times with slightly different descriptions, data preprocessing was required to ensure consistency.

Data processing, feature engineering, model training and evaluation were performed in Google Colab. This cloud-based Jupyter environment provided the computational power necessary for handling large datasets and parallelising ensemble models, utilising libraries such as Pandas, NumPy, Scikit-learn, XGBoost, SciPy, Matplotlib, and Joblib.

The dataset was initially cleaned by removing duplicate records and filtering out invalid entries, particularly those with missing or zero completion times. Additionally, outliers were excluded by removing records where the actual production time exceeded the 99th percentile of the technological completion time distribution. Figure 2 shows the code snippet used to clean the data in Google Colab. This reduced the dataset from 157,312 to 64,780 records.

To prepare the dataset for modelling, both the textual and numerical attributes were transformed into a consistent numerical representation suitable for machine learning algorithms.

First, the textual descriptions of production operations were vectorised using the TF-IDF (term frequency–inverse document frequency) method, limiting the dimensionality to 300 features. This technique assigns higher weights to words that occur frequently within a given description of a production operation, but less frequently across all descriptions. This allows the model to capture meaningful differences between types of operation.

As shown in Figure 3, inspection of the resulting TF-IDF matrix revealed that the Polish term *gratować* (meaning '*to deburr*') was the most prevalent term in the dataset, reflecting its high frequency in the operation descriptions.

Data cleaning, removing duplicates, removing rows with zero or missing times, removing extremes

```
[3]  ✓ 0s
print(f"The dataset contains {len(df)} records.")
The dataset contains 157312 records.

[4]  ✓ 0s
df = df.drop_duplicates()

[5]  ✓ 0s
df = df[(df["ProductionCompletionTime"] > 0) & (df["TechnologicalCompletionTime"] > 0)]

[6]  ✓ 0s
df = df[df["ProductionCompletionTime"] < df["TechnologicalCompletionTime"].quantile(0.99)]

[7]  ✓ 0s
df = df.dropna(subset=["Description"])

[8]  ✓ 0s
print(f"After cleaning, {len(df)} records remain.")
After cleaning, 64780 records remain.
```

Fig. 2. Data cleansing process in Google Colab, removing duplicates, invalid rows and extreme values

Feature generation

```
[9]
✓ 0s
tfidf = TfidfVectorizer(max_features=300)
X_text = tfidf.fit_transform(df["Description"].astype(str))

[12]
✓ 0s
feature_names = tfidf.get_feature_names_out()
tfidf_weights = np.asarray(X_text.mean(axis=0)).ravel()
top_words = sorted(list(zip(feature_names, tfidf_weights)), key=lambda x: -x[1])[:1]
print(top_words)

[('gratować', np.float64(0.07566360994917004))]
```

Fig. 3. Code snippet for TF-IDF vectorisation of operation descriptions in Google Colab

In the primary experiment, the categorical attribute *Operation Number* was encoded numerically using *LabelEncoder*, converting the textual identifiers into integers (Figure 4). This ensured that the machine learning models could process the categorical information without introducing an artificial order between the categories. Alternative categorical encoding strategies are examined separately as part of the supplementary analyses.

The numerical features *TechnologicalCompletionTime* and the encoded operation number were standardised using the *StandardScaler* with the parameter mean = False.

Unlike the default configuration, which centres each feature by subtracting its mean and scaling by the standard deviation, this variant performs only variance scaling. The decision to omit centring was made to preserve the non-negative structure of the features and to avoid potential numerical issues related to sparse representations. The logarithmic histograms in Figure 5 illustrate the distribution of the *TechnologicalCompletionTime* variable before and after scaling. While feature scaling does not modify the skewness of the underlying distribution or eliminate extreme values, it standardises the scale of the features, which may improve numerical stability and optimisation behaviour in models that are sensitive to feature magnitude.

After all preprocessing and transformation steps were completed, the final feature matrix was generated for model training. The resulting matrix contained 64,780 records and 302 features, where each record represented a single production operation (Figure 6). Two of these features originated from the numerical data – technological completion time and encoded operation number – while the remaining 300 features were derived from textual descriptions of operations using the TF-IDF method. This combined representation enabled the model to capture both numerical and linguistic characteristics of the production process, improving its ability to generalise patterns across different types of operations.

```
[13]
✓ 0s
le = LabelEncoder()
df["Operation_number_encoded"] = le.fit_transform(df["\ufeffOperation number"])

[14]
✓ 0s
df[["\ufeffOperation number", "Operation_number_encoded"]].head(10)
```

	Operation number	Operation_number_encoded
52	102122	0
60	102123	1
63	102123	1
66	102123	1
67	102123	1
68	102124	2
76	106027	3
80	106203	4
81	106203	4
84	106226	5

Fig. 4. Encoding of the operation number feature using *LabelEncoder*

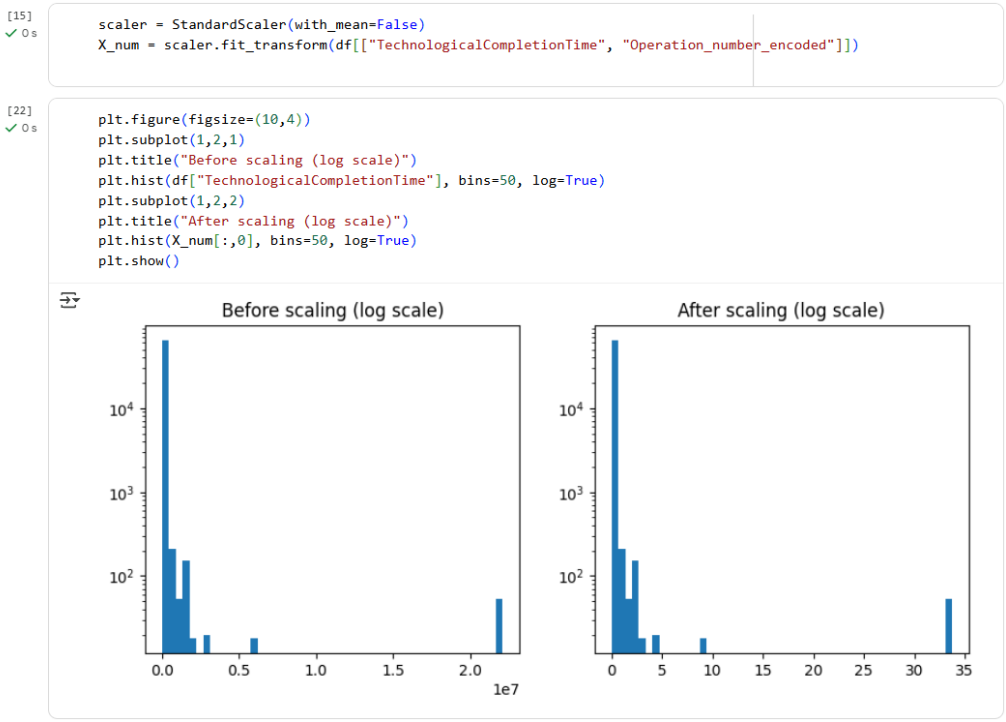

Fig. 5. Distribution of technological completion times before and after feature scaling (logarithmic scale)

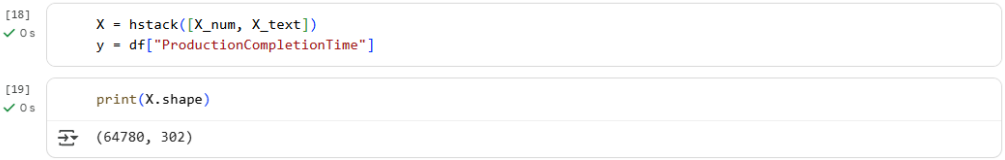

Fig. 6. Structure of the final feature matrix combines numerical and textual features

Predictive models and training procedure

Three machine learning models were evaluated in order to predict the completion time of technological operations. Linear regression was used to gain a preliminary understanding of the relationships between features and the target variable. Random Forest Regression uses an ensemble of decision trees to capture nonlinear dependencies in the data and reduce prediction variance. The XGBoost Regressor, which is based on gradient boosting, improves prediction accuracy by iteratively correcting the errors of successive trees, making it particularly effective at modelling complex relationships.

To compare the models' performance, a baseline was also established using a single feature, *TechnologicalCompletionTime*, as a simple prediction strategy.

The preprocessed dataset was split into a training set (80%) and a test set (20%) using a fixed random seed to ensure reproducibility. The regression models were implemented as Linear Regression, Random Forest with 200 trees, and XGBoost with 300 estimators and a learning rate of 0.1. Figure 7 shows parts of the Google Colab notebook where these models were defined and trained.

The performance of the models was assessed on the test set using the MAE, RMSE, and R^2 metrics. Linear regression was additionally validated using 2-fold cross-validation to assess the consistency of its performance. The reduced number of folds was chosen intentionally, as linear models are computationally efficient and their variance across folds is typically low. Furthermore, specific analyses were conducted to evaluate the impact of different encoding techniques, given that linear algorithms are particularly sensitive to the representation of categorical variables.

In contrast, the ensemble methods (Random Forest and XGBoost) were evaluated using a fixed train–test split. To bolster the robustness of the experimental design, a hyperparameter sensitivity analysis was performed for both models. This process enabled the iden-

Model training

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
baseline_feature = df["TechnologicalCompletionTime"].values.reshape(-1, 1)
X_train_bl, X_test_bl, y_train_bl, y_test_bl = train_test_split(
    baseline_feature, y, test_size=0.2, random_state=42)
bl_model = LinearRegression()
bl_model.fit(X_train_bl, y_train_bl)
baseline_pred = bl_model.predict(X_test_bl)

baseline_mae = mean_absolute_error(y_test_bl, baseline_pred)
baseline_rmse = np.sqrt(mean_squared_error(y_test_bl, baseline_pred))
baseline_r2 = r2_score(y_test_bl, baseline_pred)

print("\nBaseline (LinearRegression on TechnologicalCompletionTime):")
print(f"MAE  = {baseline_mae:.4f}, RMSE = {baseline_rmse:.4f}, R² = {baseline_r2:.4f}")
```

```
Baseline (LinearRegression on TechnologicalCompletionTime):
MAE  = 9832.2995, RMSE = 21984.0221, R² = 0.0027
```

```
models = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(n_estimators=200, random_state=42),
    "XGBoost": XGBRegressor(n_estimators=300, learning_rate=0.1, random_state=42)}
```

```
Model training:   0%|          | 0/3 [00:00<?, ?it/s]
Model: Linear Regression
Model training:  33%|███       | 1/3 [00:03<00:06,  3.06s/it]Linear Regression - MAE: 7892.13, RMSE: 20218.36, R²: 0.1565

Model: Random Forest
Model training:  67%|██████    | 2/3 [11:34<06:47, 407.89s/it]Random Forest - MAE: 4950.16, RMSE: 17959.51, R²: 0.3344

Model: XGBoost
Model training: 100%|██████████| 3/3 [11:36<00:00, 232.13s/it]XGBoost - MAE: 4807.29, RMSE: 16943.31, R²: 0.4076
```

Fig. 7. Google Colab excerpts for defining, training, and evaluating the Linear Regression, Random Forest, and XGBoost models, including the baseline comparison

tification of stable configurations and ensured a balance between precision and generalisation, effectively mitigating the risks associated with a non-iterative split.

While this evaluation strategy carries a potential risk of limited sensitivity to overfitting or dataset-specific bias, the dataset's substantial size was considered sufficient to provide a stable, representative distribution.

All trained models were saved for reuse, and their results were summarised in tables and visualised to enable clear comparison of predictive performance.

Supplementary analyses

In addition to the primary experimental study, supplementary analyses were performed to provide a more detailed understanding of model robustness and the sensitivity of predictive performance to hyperparameters and feature representation. These analyses focused on two separate aspects.

First, dedicated hyperparameter sensitivity analyses were carried out for the Random Forest and XGBoost models in isolation. Each analysis systematically evaluated the impact of key hyperparameters on predictive performance. For Random Forest, these were the number of estimators and the maximum tree depth; for XGBoost, the learning rate. Then, performance evaluation metrics, including MAE, RMSE, and R^2, were calculated for each hyperparameter combination. The results are visualised in Figures 8 and 9. These figures reveal trends in model performance across different parameter settings and pinpoint configurations that balance accuracy and stability.

Secondly, a comparative analysis of categorical feature encoding methods was conducted for linear regression models. To isolate the influence of encoding strategies, the analysis was restricted to a reduced set of features consisting of numerical process attributes and a categorical operation identifier. The textual features based on TF-IDF, which were used in the primary experiment, were deliberately excluded. The

Random Forest

	n_estimators	max_depth	MAE	RMSE	R2
0	50	nan	5010.886894	17878.621247	0.340425
1	50	10.000000	5178.761592	17367.532251	0.377596
2	50	30.000000	5003.484303	17863.319482	0.341554
3	200	nan	5021.713296	17923.343811	0.337121
4	200	10.000000	5175.920035	17362.862120	0.377931
5	200	30.000000	5019.171949	17924.666282	0.337024
6	500	nan	5011.940675	17899.710949	0.338868
7	500	10.000000	5165.939405	17320.778489	0.380943
8	500	30.000000	5008.570149	17895.194755	0.339202

Fig. 8. Random Forest regression metrics for various n-estimators and max depth

XGBoost

	n_estimators	max_depth	learning_rate	MAE	RMSE	R2
0	100	nan	0.050000	5267.655273	17372.714238	0.377225
1	100	nan	0.100000	5135.866699	17452.701797	0.371477
2	100	nan	0.200000	4977.977539	17450.860838	0.371610
3	100	6.000000	0.050000	5267.655273	17372.714238	0.377225
4	100	6.000000	0.100000	5135.866699	17452.701797	0.371477
5	100	6.000000	0.200000	4977.977539	17450.860838	0.371610
6	100	10.000000	0.050000	5036.992188	17524.249713	0.366313
7	100	10.000000	0.100000	4872.573730	17676.825054	0.355231
8	100	10.000000	0.200000	4812.337891	18011.151213	0.330611
9	300	nan	0.050000	5049.064941	17307.967183	0.381858
10	300	nan	0.100000	4890.939941	17454.934890	0.371316
11	300	nan	0.200000	4843.199707	17642.293728	0.357747
12	300	6.000000	0.050000	5049.064941	17307.967183	0.381858
13	300	6.000000	0.100000	4890.939941	17454.934890	0.371316
14	300	6.000000	0.200000	4843.199707	17642.293728	0.357747
15	300	10.000000	0.050000	4826.162109	17790.983784	0.346876
16	300	10.000000	0.100000	4828.581055	18031.063862	0.329130
17	300	10.000000	0.200000	4916.025879	18226.303630	0.314523
18	600	nan	0.050000	4923.369629	17420.111596	0.373822
19	600	nan	0.100000	4824.944336	17624.775289	0.359022
20	600	nan	0.200000	4834.801758	17880.763295	0.340267
21	600	6.000000	0.050000	4923.369629	17420.111596	0.373822
22	600	6.000000	0.100000	4824.944336	17624.775289	0.359022
23	600	6.000000	0.200000	4834.801758	17880.763295	0.340267
24	600	10.000000	0.050000	4826.505859	18011.816566	0.330561
25	600	10.000000	0.100000	4897.408203	18152.414275	0.320069
26	600	10.000000	0.200000	4999.198242	18283.240851	0.310233

Fig. 9. XGBoost regression metrics for various n-estimators and max depth

performance metrics for each encoding method are visualised in Figure 10, enabling direct comparison of their effects. These results should be interpreted as exploratory and complementary, providing insight into encoding choices, but not as an alternative to the primary predictive models.

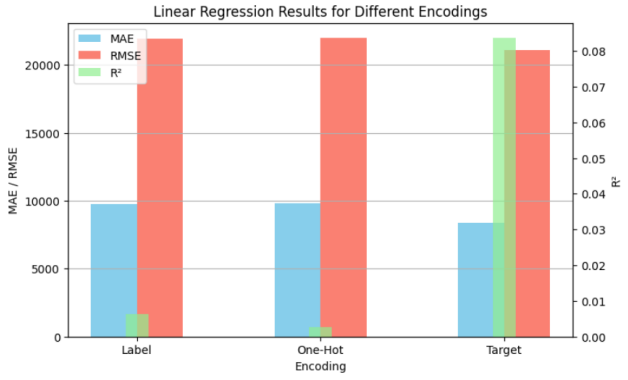

Fig. 10. Linear regression performance metrics for Label, One-Hot, and Target encoding methods

These supplementary analyses extend the primary study by evaluating the robustness of hyperparameters and the sensitivity of feature representations. They also offer a visual and quantitative reference for selecting model configurations and encoding strategies in future predictive modelling efforts.

Results and discussion

The results of the main experiment, including all developed models and the baseline, are presented in tabular and graphical formats to facilitate a direct comparison of their performance and the identification of the optimal predictive approach.

Figure 11 presents the summary of model evaluation metrics. Among all models, XGBoost achieved the best overall performance, obtaining the lowest MAE ($\approx 4{,}807$) and RMSE ($\approx 16{,}943$) values, as well as the highest coefficient of determination ($R^2 = 0.4076$). The Random Forest model performed slightly worse (MAE $\approx 4{,}950$, $R^2 = 0.3344$), while the Linear Regression achieved noticeably lower accuracy ($R^2 = 0.1564$). The baseline model, which directly used the TechnologicalCompletionTime feature as the prediction, showed by far the weakest performance ($R^2 = 0.0027$), confirming the need for a more advanced predictive approach.

Figure 12 shows scatter plots comparing actual and predicted production completion times. The diagonal red line represents perfect predictions. The XGBoost and Random Forest models exhibit predictions that cluster more closely around the diagonal, indicating a better alignment between actual and predicted values. In contrast, Linear Regression shows a larger error spread, and the baseline model clearly fails to capture the data's variability, with predictions deviating substantially from the ideal line.

Figure 13 compares the MAE and RMSE values for all models. The bar plots clearly highlight the significant error reduction achieved by the machine learning models relative to the baseline. The lowest error values observed for XGBoost confirm its superior generalisation ability and robustness on the test set.

To analyse the accuracy of the individual prediction models in relation to the actual production lead times, a comparison table was prepared. This compared the actual values (*Actual*) with the predictions of three regression models: Linear Regression, Random Forest, and XGBoost.

The baseline prediction was removed from the table to focus solely on the learning models.

Ten observations were randomly selected to facilitate visual interpretation of the results. Each row represents a single observation in the test set.

The Actual column shows the actual lead time, and the remaining columns show the predicted values by each model.

Results

```
results_df = pd.DataFrame(results, columns=["Model", "MAE", "RMSE", "R²", "CV_MAE"])
display(
    results_df.drop(columns=["CV_MAE"])
              .style
              .highlight_min(subset=["MAE", "RMSE"], color='lightgreen')
              .highlight_max(subset=["R²"], color='lightgreen')
)
```

	Model	MAE	RMSE	R²
0	Baseline (TechTime)	9832.299530	21984.022062	0.002736
1	Linear Regression	7892.134755	20218.355270	0.156495
2	Random Forest	4950.157570	17959.514325	0.334443
3	XGBoost	4807.285645	16943.311601	0.407631

Fig. 11. Summary of model evaluation metrics (MAE, RMSE, and R^2) for all regression models and the baseline

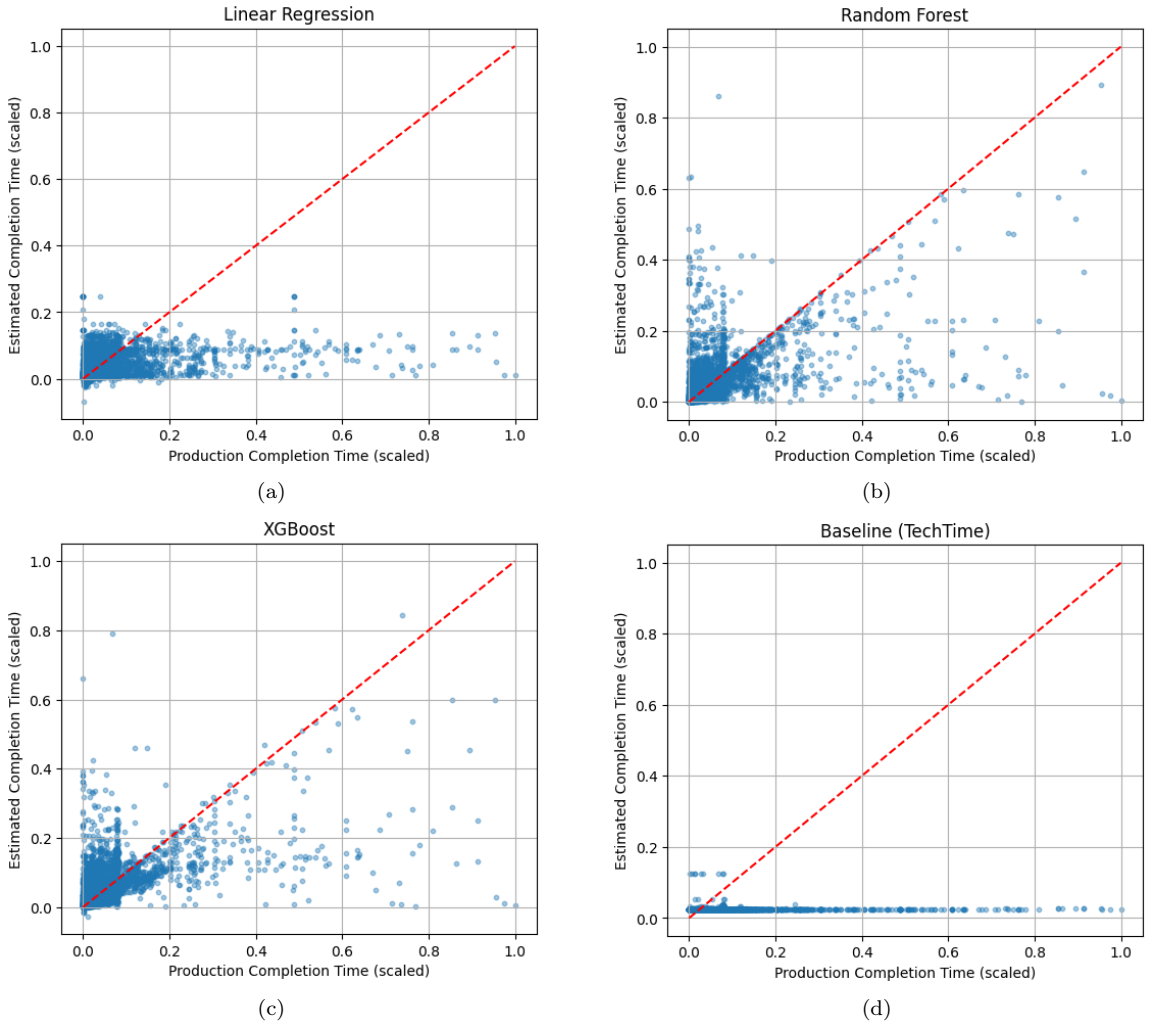

Fig. 12. Scatter plots comparing actual and predicted production completion times for each regression model and the baseline

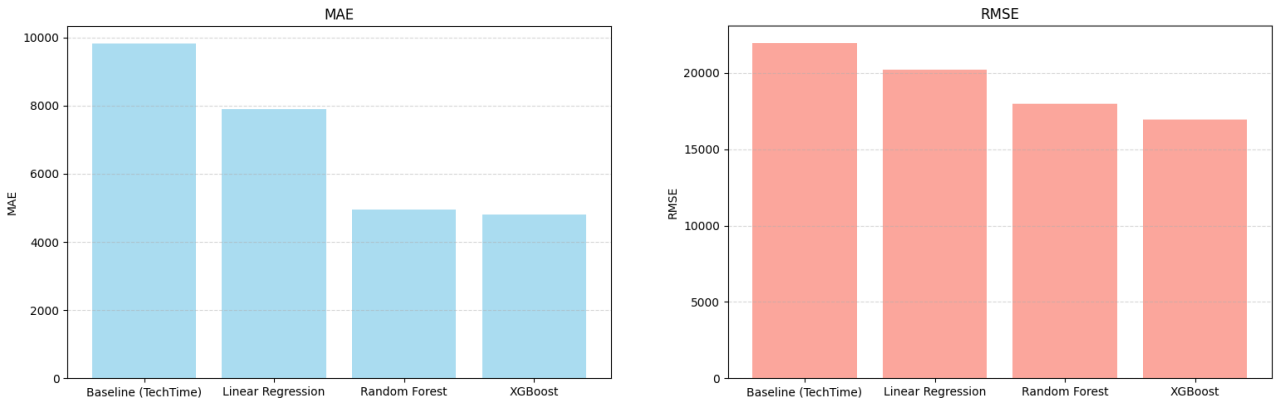

Fig. 13. Comparison of model error values for all regression models and the baseline

The prediction error of each model relative to the actual value was calculated for each observation, and the best prediction in each row was highlighted in light green. This makes it easy to see which model came closest to the actual result in each case.

Figure 14 shows an example of ten observations. Analysis of the highlighted results shows that XGBoost achieves the best predictions in most cases, suggesting its advantage in accurately representing real production times compared to other models.

	Actual	Linear Regression	Random Forest	XGBoost
82279	1020	4048.519111	1350.002500	1287.256836
91097	9000	34714.923519	28601.352143	28741.523438
90787	460	1385.952818	763.085143	768.981079
149757	13269	3376.413730	19762.866256	15506.812500
34452	2700	5267.939958	3104.296500	2883.303223
69592	4200	4203.243249	3367.604571	3589.164551
25968	6319	7967.758753	7760.657500	6713.686035
16655	780	3427.653029	2833.502167	1249.808716
65507	4800	7656.981977	3887.100000	4949.431152
133552	7200	8595.629074	3743.503583	6638.743164

Fig. 14. Comparison of actual and predicted production completion times for a sample of ten test observations, with the most accurate predictions highlighted

Although the main experiment identified XGBoost as the top-performing model, further diagnostic analyses were conducted to investigate whether specific adjustments, such as encoding techniques and hyperparameter configurations, could enhance the model's performance and stability.

The hyperparameter sensitivity analysis shows that the performance of Random Forest improves significantly when the number of trees is increased from low values, but further increases lead to diminishing performance gains. Consequently, the main experimental configuration can be considered representative.

Regarding the XGBoost model, the sensitivity analysis indicates that its performance is particularly influenced by the interaction between tree depth and the learning rate. Additional tests suggest that, while a higher learning rate or increased depth can lead to a lower mean absolute error (MAE), these changes do not always result in a higher R^2. Specifically, exceeding a depth of six occasionally resulted in a slight decline in generalisation, suggesting that the parameters chosen for the main experiment are within a stable and effective range for this dataset.

Furthermore, the encoding comparison highlights the sensitivity of linear regression models to the representation of high-cardinality categorical variables. While target encoding improves linear regression performance relative to label and one-hot encoding in a reduced-feature setting, its predictive accuracy remains substantially lower than that of the ensemble tree-based models evaluated in the primary experiment. This observation further supports the selection of nonlinear ensemble methods for modelling complex production processes.

Conclusions

A comparative analysis of regression models revealed that tree-based algorithms, such as Random Forest and XGBoost, outperformed linear regression in forecast accuracy for production lead time. XGBoost in particular achieved the lowest prediction error relative to actual values in most cases, confirming its effectiveness in representing complex data dependencies. Baseline prediction was significantly less accurate, highlighting the superiority of machine learning models. These findings were reinforced by supplementary analysis

showing that ensemble models perform consistently well across a range of hyperparameter configurations and data representations.

Machine learning is currently a crucial tool for analysing and forecasting industrial processes more broadly. Thanks to their ability to detect patterns in large, nonlinear datasets, machine learning algorithms can make more accurate predictions and support decision-making and resource optimisation. The results of this analysis confirm that ensemble models such as XGBoost can effectively cope with data variability and the nonlinearity of production processes. In practice, such predictive models can be implemented in manufacturing planning systems to dynamically estimate production completion times and identify potential delays before they occur. This leads to improved scheduling efficiency, reduced downtime, and overall optimisation of production workflows. Further sensitivity studies suggest that these models can maintain reliable performance even in environments with limited parameter-tuning opportunities. This is particularly relevant for real-world industrial applications.

Directions for further development include exploring new ensemble algorithms to improve prediction accuracy. Further research could also focus on integrating these models into real-time decision support systems to enable dynamic adjustments to production schedules and improve resource management. Furthermore, the presented diagnostic framework should be extended in future work by incorporating automated hyperparameter optimisation and systematically evaluating feature encoding strategies for hybrid models that combine numerical, categorical and textual data sources. These analyses could provide a deeper understanding of model interpretability and long-term stability in dynamic production environments.

References

Ampuła, D. (2017). The Power of Prediction Process in the Tests of Ammunition Elements. *Journal of Konbin*, 43(1), 23-35. DOI: 10.1515/jok-2017-0038.

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* (pp. 785–794). ACM. DOI: 10.1145/2939672.2939785

Drzewiecki, P. (2017). Predykcja poziomu tlenu w piecu EAF z wykorzystaniem metod sztucznej inteligencji: regresji liniowej i metody najbliższych sąsiadów (k-NN). *Hutnik-Wiadomości Hutnicze, 84*(2), 79–83.

Grandhi, B.S. (2019). *Predictive modelling using machine learning techniques for railway incident based parameters - Denmark Railways* (Master's thesis). Technical University of Munich.

Kumar, V., & Garg, M.L. (2018). Predictive analytics: A review of trends and techniques. *International Journal of Computer Applications, 182*(1), 1–4. DOI: 10.5120/ijca2018917434

Lamba, M., & Madhusudhan, M. (2022). Predictive modeling. In M. Lamba & M. Madhusudhan (Eds.), *Text mining for information professionals* (pp. 213–242). Springer Nature. DOI: 10.1007/978-3-030-85085-2_9

Miller, C., Portlock, T., Nyaga, D.M., & O'Sullivan, J.M. (2024). A review of model evaluation metrics for machine learning in genetics and genomics. *Frontiers in Bioinformatics, 4*, 1–13. DOI: 10.3389/fbinf.2024.1457619

Plevris, V., Solorzano, G., Bakas, N.P., & Ben Seghier, M.E.A. (2022). Investigation of performance metrics in regression analysis and machine learning-based prediction models. *Proceedings of the 8th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2022)*. DOI: 10.23967/eccomas.2022.155

Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports, 14*, Article 6917. DOI: 10.1038/s41598-024-56706-x

Sobaszek, Ł., Gola, A., & Świc, A. (2017). *Predictive Scheduling as a Part of Intelligent Job Scheduling System* (pp. 358–367). Springer, Cham. DOI: 10.1007/978-3-319-64465-3_35

Tober, S. (2020). *Tree-based machine learning models with applications in insurance frequency modelling* (Bachelor's thesis). KTH Royal Institute of Technology.

Wójcik, F. (2018). Prognozowanie dziennych obrotów przedsiębiorstwa za pomocą algorytmu XGBoost – studium przypadku. *Studia Ekonomiczne. Zeszyty Naukowe Uniwersytetu Ekonomicznego w Katowicach, 375*(14), 121–140.