sensors

MDPI

Article

Predictive Maintenance Using Administrative Work Orders: A Machine Learning Framework for Failure Criticality Classification

Juan Cruz Martinez[1], Alejandro Garcia Rodriguez[2]

[1] Facultad de Ingeniería Mecánica, Universidad Santo Tomás, Carrera 9 No. 51-11, Bogotá, Colombia
* Correspondence: juanacruz@usantotomas.edu.co.

Abstract

This study evaluates the feasibility of predictive maintenance using only historical corrective maintenance records from a food industry plant, without sensor data. The dataset comprised 6,731 failure events across 664 machines, with nine operational variables. Failure criticality was classified into low (≤23 min), medium (24–44 min), and high (>44 min) based on downtime percentiles. Three tree-based models (Random Forest, XGBoost, LightGBM) and a multilayer perceptron were developed. Hyperparameter optimization was performed using RandomizedSearchCV and Optuna, the latter with a custom gap-penalized objective function to mitigate overfitting. A chronological split (70/15/15) respected temporal order. The best model, XGBoost with gap-penalized Optuna, achieved a test Macro F1 of 0.3829 and a weighted F1 of 0.4871, with a minimal generalization gap. The neural network obtained the highest weighted F1 (0.5512) but failed on the medium criticality class. Results confirm that administrative records alone have a structural predictive ceiling, and that explicit gap penalization is effective for boosting models. The proposed framework enables systematic comparison of tree-based and neural models for failure criticality classification using only work order logs. While statistical methods such as ANOVA can establish whether significant differences exist between operational groups, they do not provide a predictive framework capable of classifying new failure events at the moment they occur. Conversely, machine learning models are designed to identify and generalize patterns from historical data, but do not inherently explain the statistical structure of the underlying variables. This study addresses both dimensions in a complementary fashion: ANOVA and post-hoc Tukey HSD tests are used to statistically validate the relevance of machine identity, work shift interaction, and technician count on equipment downtime, while machine learning models are subsequently developed to assess whether those patterns are learnable and generalizable to unseen failure events. This dual approach enables a more complete characterization of industrial failure dynamics than either method alone.

Keywords: Machine learning; Predictive maintenance; Random Forest; XGBoost, classification models

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

Citation: To be added by editorial staff during production.

1. Introduction

In industry, unscheduled equipment downtime represents one of the most significant operational challenges, as it directly affects productivity, generates additional costs, and reduces machine availability. To address this problem, predictive maintenance has become a key strategy that, through the use of sensors, machine learning techniques, and analysis of historical failure data, makes it possible to anticipate and/or mitigate the risk of unexpected failures [1].

Proper data management and processing are essential in predictive maintenance, since applying appropriate analysis tools allows identifying patterns, correlations, and anomalous behaviors in equipment operation. In this regard, it has been highlighted that sensors integrated into IoT architectures facilitate the continuous collection of critical variables such as vibration, temperature, or pressure, but the quality of predictions depends directly on the adequate preparation and transformation of such data before its incorporation into machine learning models [2]. Complementarily, it has been pointed out that the integration of artificial intelligence, big data, and IoT has enhanced the capability of predictive maintenance by providing an environment in which industrial systems can learn from large volumes of information and dynamically adapt to changing conditions, thereby increasing the accuracy and robustness of the models [3].

A systematic mapping of the literature on predictive maintenance revealed that the most widely used algorithms were classification and regression methods, as well as hybrid approaches based on deep neural networks (LSTM, CNN), random forests, and support vector machines (SVM) [4]. These methods increase the effectiveness of predictive maintenance by improving early fault detection, estimating remaining useful life, and optimizing the allocation of maintenance resources. However, it was also noted that there is no single optimal algorithm for all problems, which forces techniques to be adapted to each particular case. Nevertheless, achieving these results requires working with reliable and representative databases capable of reflecting the real operational dynamics of the equipment. On the other hand, various machine learning (ML) models are used as tools for predictive maintenance, including artificial neural networks (ANN), support vector machines (SVM), and random forests (RF) [5]. These models process large volumes of data obtained from IoT sensors installed on equipment, learning patterns from historical operation and failure data to predict future component status, remaining useful life (RUL), or the probability of anomalies. Thanks to feature extraction and data analysis techniques, the use of ML has enabled a shift from reactive or scheduled strategies to proactive approaches, reducing unplanned downtime, optimizing resource use, and lowering operational costs.

Complementarily, it has been pointed out that the success of predictive maintenance in Industry 5.0 depends not only on the accuracy of the models but also on their acceptance and integration within the organization [6]. Generating trust and transparency in predictive systems, offering users some control over the results, and designing tools that facilitate decision-making without increasing cognitive load are key aspects. These factors are essential for transforming model predictions into effective maintenance actions in an industrial environment [6]. Likewise, several machine learning techniques have been em-ployed to detect failures and schedule maintenance before they occur. Random Forest (RF) is the main algorithm, built from a set of decision trees that improves accuracy by com-bining multiple classifiers and using a large number of trees. Decision Tree (DT) is used as a reference and shows less stability and accuracy in response to data changes. Linear Re-gression (LR) and Symbolic Regression (SR) are also evaluated as comparative metrics, in addition to the use of AutoML for automatic feature selection and parameter tuning, and the Naïve Bayes algorithm in the fine-tuning phase [7]

The objective of this research was to identify patterns and relationships in failure data that would allow anticipating equipment failures, optimizing unscheduled downtime, and analyzing the influence of operational and human factors on maintenance management. Based on the above, the methodology described in the following section was developed, grounded in exploratory analysis of historical failure data and preparation of the database for future applications of machine learning models

2. Materials and Methods

The dataset consisted of historical corrective maintenance records from industrial equipment. It contained 9 predictor variables: Week, Equipment, Line, Product, Shift, Month, Plant, Subcategory, and Failure_Type. The target variable Failure_Type had three classes. The data were split into three subsets: training 70%, validation 15 and test 15% reserved for final evaluation. All preprocessing steps including encoding, scaling, and any feature transformation were fitted exclusively on the training set and then applied to the validation and test sets to prevent data leakage [8–10].

Phase 1 – Dataset Description

The study was developed from a history of corrective maintenance failures provided by a company in the food industry. The dataset comprises 6,731 records distributed across 27 variables that capture operational, technical, and administrative information for each failure event, including downtime, affected equipment, recorded cause, and human resources assigned to the operation. Processing and analysis were carried out in the Google Colab environment using Python 3, with the libraries pandas, numpy, matplotlib, seaborn, XGBoost, and LightGBM.

Phase 2 – Data Preprocessing and Cleaning

Prior to analysis, an inspection of the dataset was performed using the df.info() and df.isnull().sum() functions to characterize the data types and quantify missing values per variable. Columns with format inconsistencies were standardized: the variable Tiempo (Min) Línea was converted to numeric type using pd.to_numeric(), the variable Trabajos Provisionales was normalized using str.strip().str.upper() to unify entries, and the variable Subcategoría was standardized with str.strip().str.title() to correct data inconsistencies. Additionally, columns with a high proportion of missing values, such as Aprobó and Ejecutó, were identified.

Phase 3 – Exploratory Data Analysis

An exploratory data analysis was conducted to characterize the historical failure behavior and identify potentially relevant variables for subsequent predictive model development. The analysis was divided into subphases. Phase 1 included the distribution of failures by equipment, plant, failure cause, work shift, and time period. Phase 2 included the calculation of total and average downtime per equipment, cause, and resource category. Phase 3 involved the identification of recurrent equipment–cause combinations through joint grouping. Phase 4 consisted of a criticality map using a scatter plot of frequency versus lost time per equipment. Phase 5 performed a distributional analysis of downtime for the ten equipment with the highest accumulated unavailability using KDE plots, histograms, shift-based boxplots, and bivariate density estimates. All visualizations were generated using the matplotlib and seaborn libraries.

Phase 4 – Inferential Statistical Analysis

To determine whether there are statistically significant differences in the average downtime across different grouping factors, four analysis of variance (ANOVA) tests were applied. The first ANOVA evaluated the effect of equipment on downtime (one factor), testing the null hypothesis that the average downtime is equal for all equipment against the alternative that at least one equipment has a different average. The second ANOVA evaluated the equipment × work shift interaction on the same subset (two factors), checking whether the average downtime changes according to equipment, according to shift, or whether the equipment–shift combination produces significant differences. The third ANOVA evaluated the effect of the number of assigned technicians on downtime (one factor), testing whether the average downtime is equal for all numbers of assigned technicians or whether at least one number yields a different average.

In all cases, Tukey's HSD post-hoc test was applied using the pairwise_tukeyhsd() function from the statsmodels library, with a significance level of $\alpha = 0.05$. ANOVA was selected as the contrast method because it is robust for multiple comparisons and widely validated in industrial maintenance studies [8].

Phase 5 – Machine Learning

Data source: The study used the historical corrective work order records from a food manufacturing plant in Colombia, covering the period December 2023 to December 2024. The original dataset comprised 6,731 records and 27 columns. No sensor data were used: all variables derive exclusively from administrative corrective maintenance records.

Fault description standardization: The column Fault Description and/or Cause presented 5,336 distinct written forms across the 6,731 records, arising from the absence of a controlled vocabulary during fault registration: each operator described the same event using abbreviations, spelling errors, and variable terminology. This fragmentation made direct incorporation of the text column as a predictive variable unfeasible. To address this problem, a keyword-matching standardization process was designed that transformed the descriptions into 73 standard categories called Fault_Type. The strategy consisted of building a priority-ordered dictionary: more specific categories (e.g., Demolding Unit, Sonotrode) are evaluated before more general ones (e.g., Jam/Blockage, Belt), preventing misclassification of descriptions containing multiple components. The process classified 79.9% of records; the remaining 20.1% (1,352 records) were labeled as Others, corresponding to generic descriptions without sufficient information to identify the affected component[12].

The target variable Criticality did not exist in the original database and was constructed from the variable Equipment Downtime (Min), which records the duration in minutes of the equipment stoppage. A preliminary step eliminated 301 records with zero downtime — physically impossible in the context of a maintenance intervention [13] — reducing the dataset to 6,430 records. This proportion (4.5%) falls below the 5% threshold established by Han, Kamber and Pei [9] for the methodological validity of listwise deletion.

Classification thresholds were defined using the P60 and P80 percentiles of the downtime distribution: P60 = 23 minutes and P80 = 44 minutes. As additional validation, the MDLP (Minimum Description Length Principle) discretization algorithm proposed by Fayyad and Irani [14] was applied, identifying a first cut at 23.5 minutes — validating the lower threshold. The P80 upper threshold was preferred over the second MDLP cut (28.5 min) to ensure sufficient statistical representativeness in the medium class and for its alignment with the Pareto principle: the top 20% of faults concentrate the largest share of availability losses. The resulting distribution is presented in table 1

Table 1. Distribution of the target variable criticality

Class	Time range	Records	Percentage
Low	1 – 23 min	3884	60.4%
Medium	24 – 44 min	1271	19.8%
High	>44 min	1275	19.8%
Total		6430	100%

Thresholds defined by P60 and P80 percentiles of the equipment downtime distribution.

Predictive variable selection – exclusion due to data leakage: Two types of variables were identified and excluded due to data leakage. Type 1 — future information not available at the time of the fault: Solution, Responsible Technician, Suggested Comment, Approved By, Executed By, Reviewed By, Provisional Work, and Technician Count. Type 2 variables that directly contain the target variable: Equipment Downtime (Min) (source of Criticality), Line Downtime (Min), Equipment/Line Downtime (Hrs), and Greater than or equal to 30 min

Predictive variable selection – mutual information: To quantify the informational contribution of candidate variables, Mutual Information $I(X; Y) = H(X) + H(Y) - H(X, Y)$ was calculated between each predictive variable X and the target variable Y, using mutual_info_classif from scikit-learn with random_state=42. The nine resulting candidate variables are presented in Table 2.

Table 2. Mutual information of candidate variables with respect to criticality.

Variable	Mutual information	Decision
Machine	0.0623	Included
Product	0.0348	Included
Line	0.0170	Included
Fault_Type	0.0112	Included
Week	0.0059	Included
Shift	0.0042	Included
Month	0.0010	Included
Plant	0.0006	Included
Subcategory	0.0000	Included

All 9 variables were retained since variables with low individual mutual information may contribute through interactions with other variables.

Chronological Dataset split: The dataset was divided into three sets in 70%/15%/15% proportions strictly following the chronological order of records. Sivakumar et al. [15] and Xu and Goodacre [16] establish that a training proportion of 60–70% offers a reasonable balance between learning capacity and evaluation representativeness. The split was chronological — not random — since a random split would allow the model to learn from future faults during training, generating artificially optimistic estimates [17,18]. Table 3 presents the resulting distribution.

Table 3. Chronological dataset split.

SET	Proportion	Records	Start	End
Training	70%	4501	27/12/2023	30/08/2024
Validation	15%	964	30/08/2024	21/10/2024
Test	15%	965	21/10/2024	12/12/2024

The test set remained completely isolated throughout the entire training, validation, and optimization process.

Preprocessing and variable encoding: Two encoding strategies were applied according to model type. For Random Forest, XGBoost, and LightGBM, LabelEncoder was used, assigning an integer to each category. The encoder was trained exclusively on training data and subsequently applied to validation and test sets. The result was a dataset of 9 numerical columns per set.

For the neural network, One-Hot Encoding (OHE) was applied using pd.get_dummies, converting the 7 text categorical variables into 983 independent binary columns. The variables Week and Shift remained as numerical variables since they were already encoded as integers, resulting in a total of 985 input columns. Since Week (range 1–36) and Shift (range 1–3) presented scales significantly different from the 983 OHE columns (range 0–1), StandardScaler was applied to these two variables, trained exclusively on the training set to prevent data leakage.

Class imbalance handling: The 60/20/20 distribution across Low, Medium, and High classes presents an imbalance that, without correction, may lead the model to systematically predict the majority class, achieving 60% accuracy without having learned any real pattern. To compensate, the class_weight='balanced' parameter was used in Random Forest and LightGBM, which automatically adjusts each class weight. For XGBoost which does not directly accept class_weight in multiclass classification the sample_weight parameter was used, assigning the corresponding weight to each individual training record, achieving the same effect.

3. Results

3.1. General characterization of the failure History

The failure history analyzed comprises 6,731 records of corrective maintenance events across 664 equipment units. The initial inspection of the dataset confirmed that the most analytically relevant variables (Equipment, Line, Time (Min) Equipment, Number of Technicians, Resource Category, and Subcategory) were completely populated, ensuring data availability for the analysis (Table 4). The column Approved presented the highest percentage of missing values with 6,630 nulls, followed by Solution with 257 nulls, executed with 186 nulls, and reviewed with 178 nulls. These variables were identified for exclusion from the predictive model, since their high proportion of missing values can degrade model performance and must be either cleaned or removed during preprocessing, in accordance with data quality criteria for stable supervised classification models [9].

Table 4. Number of missing values per variable

Variable	Number of missing values
Week	0
Date	0
Time (Min) Equipment	0
Time (Min) Line	0
Code	0
Equipment	0
Line	0
Product	0
Shift	7
Origin	0
Description and/or cause of failure	5
Solution	257

Approved	6630
Executed	186
Report number	0
Reviewed	178
Provisional jobs	0
Month	2
Time (Hrs) Equipment	0
Time (Hrs) Line	1
Responsible Technician	0
Plant	0
Number of Technicians	0
Greater than or equal to 30 Min	0
Category by resource	0
Suggested comment	0
Subcategory	0

3.2. Distribution of Failures by Equipment

The analysis of failure frequency by machine revealed a distribution highly concentrated in a small subset of assets (Figure 1 - a). Machine 0078 recorded the highest number of events with 258 failures, followed by machine 0041 with 163 events, and machine 0114 with 127 recorded events. Together, the ten machines with the highest failure frequency accumulated 19.31% of all events in the history, despite representing a minimal fraction of the total asset base. This behavior is consistent with the Pareto principle applied to industrial maintenance systems, according to which approximately 20% of machines generate 80% of corrective interventions, a principle widely documented and used as a prioritization criterion in asset management [10].

3.3. Distribution of Failures by Accumulated downtime

The analysis of total downtime per machine revealed a substantially different ranking than that of failure frequency, evidencing that assets with the highest number of events are not necessarily those with the greatest impact in terms of accumulated unavailability (Figure 1 - b). Machine 0123 led this ranking with 9,634 minutes of accumulated downtime, followed by machine 0202 with 7,992 minutes and machine 0189 with 6,775 minutes. Machine 0078, which led the frequency ranking with 258 events, ranked sixth in accumulated downtime with 6,296 minutes, indicating that its failures are numerous but relatively quick to resolve. This dissociation between frequency and lost time reinforces the need to consider the operational impact of failures when prioritizing assets. According to the RCM II methodology, maintenance decisions should not be based solely on failure occurrence frequency, but rather on the consequences that failures generate on operation, safety, and system costs [11].

3.4. Analysis by Production Plant

The distribution of failures by production plant showed that Plant 1 concentrated the highest frequency of events with 2,464 records, followed by Plant 4 with 1,637 records and Plant 2 with 1,096 records. However, in terms of total lost time, Plant 4 surpassed Plant 1. This reversal in the ranking between frequency and accumulated time by plant is consistent with the finding from the previous section: failure frequency is not a sufficient indicator of the real operational impact (Figures 1 - c and 1 - d). Plants 3 and 5 presented the lowest values in both metrics, which could reflect differences in machine age, production load, or maintenance practices at each facility.

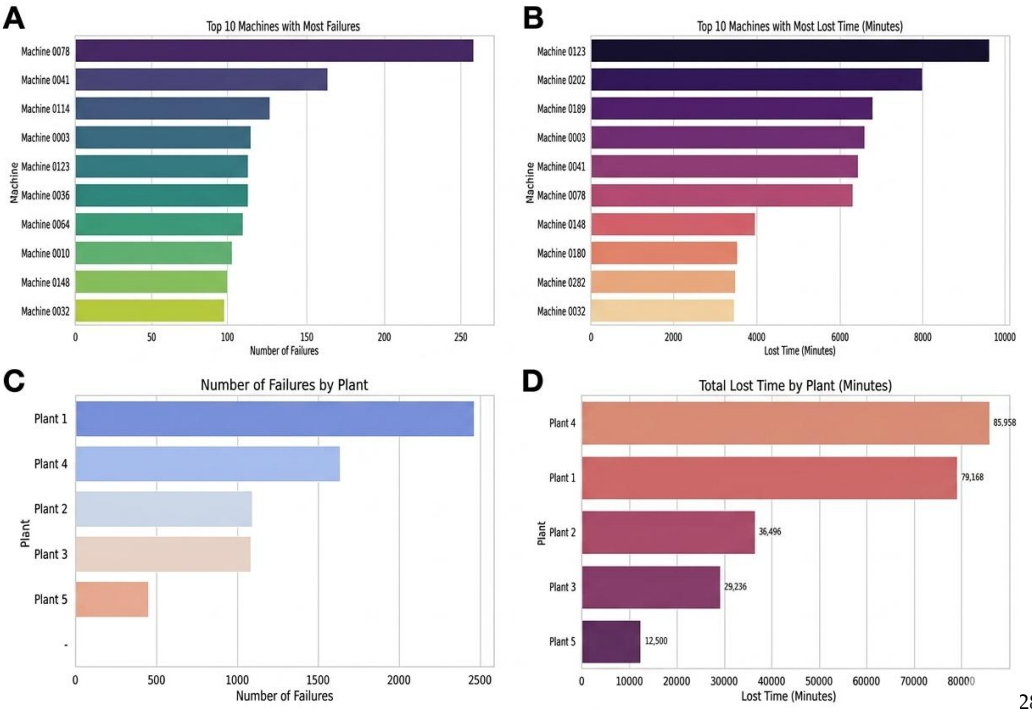

Figure 1. Failure analysis summary: (**a**) Machine with most failures; (**b**) Machine with most lost time; (**c**) Number of failures by plant; (**d**) Total lost time by plant

3.5. Temporal Analysis: Monthly Evolution of Lost Time

The monthly distribution of total lost time, as shown in Table 5, revealed significant variability over time. April recorded the highest accumulated value with 29,215 minutes of downtime, followed by January with 27,252 minutes and March with 25,771 minutes. The months with the lowest impact were December with 8,295 minutes and August with 15,555 minutes. The observed monthly variability suggests the existence of operational patterns associated with production cycles, availability of technical personnel, or periods of high demand — factors that have been identified in the literature as determinants of maintenance workload.

Table 5. Monthly evolution of total downtime (in minutes).

MONTH	TIME(MIN)MACHINE
APRIL	29215
JANUARY	27252
MARCH	25771
MAY	22894
FEBRUARY	22393
OCTOBER	22134
JUNE	19875
JULY	16959
SEPTEMBER	16742
NOVEMBER	16245
AUGUST	15555
DECEMBER	8295

3.6. Criticality Map

The criticality map presented (Figure 2 - a) allowed the visual classification of assets into four operational profiles. Machine 0123 was positioned in the highest criticality zone. Machines 0003, 0041, and 0078 formed a second high-criticality group due to their combination of frequency and lost time. The majority of the remaining machines were grouped in the low criticality zone.

3.7. Heat Map

The quarterly heatmap of the ten critical machines (Figure 2 - b) represents the evolution of total accumulated downtime in minutes over five quarters between 2023Q4 and 2024Q4. The heatmap representation allows simultaneous identification of temporal concentration patterns, degradation trends, and anomalous behaviors across multiple assets.

Machine 0123 recorded the highest value in the dataset in the 2024Q1 quarter, experiencing a sharp decline towards 2024Q2, 2024Q3, and 2024Q4. This pattern of a sharp peak followed by a sustained reduction may be associated with the execution of a major corrective intervention during or immediately after the first quarter of 2024, which would have partially restored the availability of the asset. Machine 0003 presented an even more marked behavior in 2024Q1 and zero values in the 2024Q3 and 2024Q4 quarters, which could indicate a replacement of the asset or a major maintenance intervention that permanently eliminated the source of the failure.

Overall, the quarterly heatmap analysis revealed that none of the ten critical machines exhibit uniform temporal behavior, evidencing the coexistence of multiple failure modes with distinct temporal dynamics within the same production system.

3.8. Box plot analysis by work shift

The boxplot analysis segmented by work shift (Figure 2 – c) showed that machines 0189, 0202, and 0123 presented the highest median downtime values, with greater variability in shift 3. Machine 0189 exhibited the highest dispersion in shift 2, with multiple outliers exceeding 300 minutes, indicating the presence of high-severity failure events concentrated during that shift. Machines 0003 and 0032 presented the narrowest boxes. The outliers identified in this analysis constitute events of high informational value for the predictive model, since, as noted by Montgomery, these points can be more revealing than the rest of the data as they represent critical scenarios [8].

3.9. Bivariate Density Analysis: Assigned Technicians vs. Downtime

The bivariate kernel density (figure 2 – d) plot between the number of assigned technicians and downtime showed that the highest concentration of interventions corresponds to 2 or 3 technicians and downtime between 0 and 100 minutes. Atypical cases were observed with downtime exceeding 300 minutes for groups of 3 or more technicians, evidencing that the complexity of certain failures exceeds the capacity for rapid resolution regardless of the available human resources.

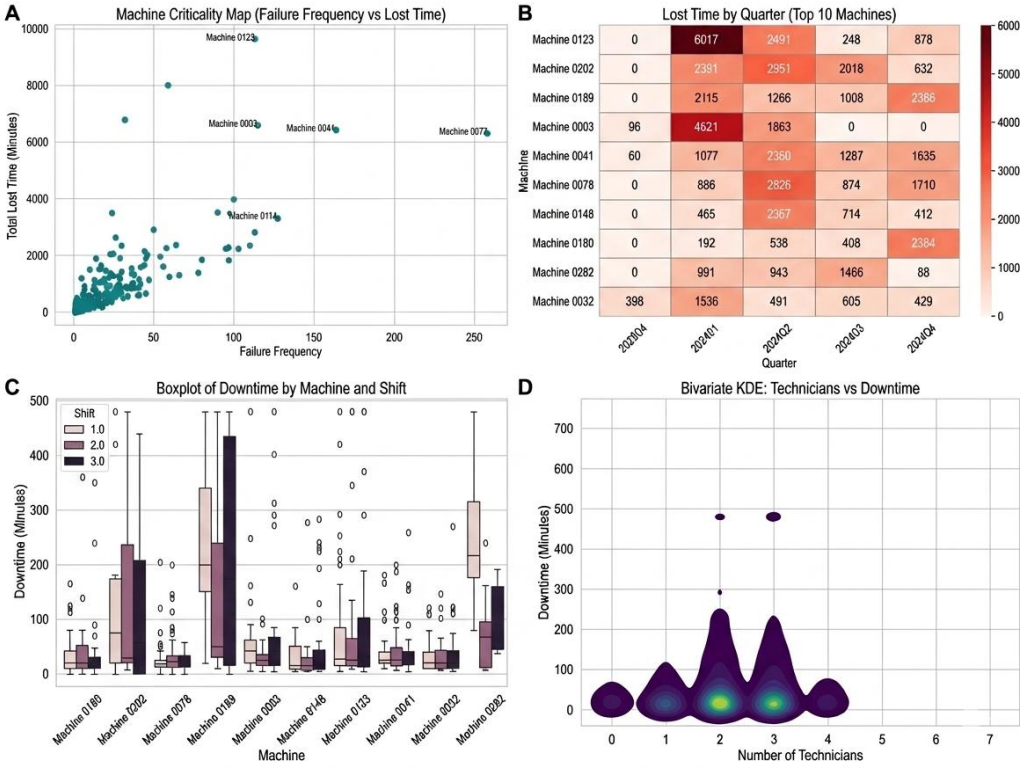

Figure 2. Operational analysis of critical machines: (**a**)criticality map showing frequency vs. lost time; (b) quarterly heatmap of total downtime for the ten most critical machines (2023Q4–2024Q4); (c) boxplot distribution of downtime by work shift for selected machines; (d) bivariate kernel density estimate of assigned technicians versus downtime.

3.10. Results of Analysis of Variance (ANOVA) and Tukey HSD Post-Hoc Tests

3.10.1 One-Factor ANOVA: Effect of Machine on Downtime

The application of Tukey's HSD post-hoc test to the complete set of machines generated comparisons with statistically significant differences (reject = True, $p < 0.05$). This result confirms that machine is a significant factor in explaining downtime. The machines that presented significantly higher downtime were machines 0001, 0030, 0057, 0066, 0123, and 0189. These results statistically validate the findings of the exploratory analysis and justify the inclusion of the machine variable as a predictor in the criticality classification model.

3.10.2. Two-way ANOVA: Machine x Work Shift

The Tukey HSD post-hoc analysis for the machine factor within the Top 10 × Shift subset confirmed that machine 0123 presents significantly higher average downtime, showing significant differences against all analyzed machines, with increases ranging from 45 to 64 minutes. In contrast, machine 0003 recorded the lowest average downtime within the analyzed subset, with average differences of 30 to 36 minutes.

For the work shift factor, the post-hoc analysis did not show statistically significant differences among the three operation shifts. These results indicate that the operating schedule does not constitute a determining factor in machine downtime within the analyzed subset.

3.10.3 One-factor ANOVA: Number of technicians vs Downtime

The Tukey HSD post-hoc analysis among the groups defined by the number of assigned technicians identified multiple comparisons with statistically significant differences. The group with 1 technician presented significantly lower downtime than the

groups with 6 and 7 assigned technicians ($p < 0.05$ in all cases). These results statistically confirm that the number of technicians may be an indicator of failure complexity, being consistent with the Maintenance Man-Hours Per Failure (MMPH) indicator proposed by Wireman [10].

3.11. Machine learning

3.11.1 Baseline Model Performance and Overfitting Analysis

Three tree-based classifiers Random Forest (RF)[23], XGBoost [20], and LightGBM [21] were first trained using default hyperparameters with class weighting to address the 60/20/20 class imbalance. Class weights were computed as wc = ntotal / (3 × nc), yielding weights of 1.59 for High and Medium classes and 0.57 for Low, forcing the models to penalize minority-class errors more heavily. The baseline Macro F1-scores on the validation set were 0.3135 (RF), 0.3632 (XGBoost), and 0.3804 (LightGBM).

A critical finding at this stage was the presence of severe overfitting across all three models. The gap reached 0.6640 for RF, 0.5388 for XGBoost, and 0.4179 for LightGBM. These results confirm that default configurations led the models to memorize training patterns rather than learn generalizable representations, motivating a systematic regularization and hyperparameter optimization process. Although the literature does not show the exact range for a gap to be good or critical, R. Ballester [22] shows that a low gap is better.

3.11.2 Hyperparameter Optimization Strategy

Due to the high gaps observed in the baseline models, the optimization strategy incorporated regularization as a primary objective alongside predictive performance. Four progressive strategies were evaluated for each tree-based model: (1) manual regularization by systematic exploration of complexity-controlling parameters [24]; (2) RandomizedSearchCV [19] with 50–100 random iterations and five-fold StratifiedKFold cross-validation; (3) Optuna [25] with standard F1 Macro maximization; and (4) Optuna with a gap-penalized objective. StratifiedKFold was chosen over TimeSeriesSplit for internal cross-validation because the class imbalance (60/20/20) required guaranteed representation of minority classes in each fold

Table 6. Macro F1-score and generalization gap (F1_train − F1_val) for each optimization strategy.

Strategy	F1 VAL RF	Gap Val RF	F1 Val XGBoost	Gap Val XGBoost	F1 Val LightGBM	Gap Val LightGBM
Baseline	0.3135	0.6640	0.3632	0.5388	0.3804	0.4179
Manual regularization	0.4070	0.2967	0.3898	0.0523	0.3944	0.2224
RandomizedSearchCV	0.3843	0.4534	0.3920	0.1607	0.3830	0.2068
Optuna (Standard)	0.4002	0.3932	0.3797	0.1483	0.3957	0.4679
Optuna + pen.Gap	0.3870	0.0702	0.4125	0.0301	0.4065	0.0661

The gap-penalized Optuna strategy produced the strongest overall results. XGBoost achieved an F1 Macro of 0.4125 on the validation set with a near-zero gap of 0.0301. LightGBM reached 0.4065 (gap: 0.0661) and RF 0.3870 (gap: 0.0702). For RF, manual regularization (max_depth = unrestricted; min_samples_leaf = 10) yielded a higher F1 Val (0.4070) with a gap (0.2967). Standard Optuna, without gap penalization, produced comparable F1 Val scores but reopened the generalization gap particularly for LightGBM (gap: 0.4679).

3.11.3 Neural network results

A multilayer perceptron (MLP) was included as a fourth model. The architecture consisted of four hidden layers (512→256→128→64 neurons) with ReLU activations, Batch Normalization, and Dropout (rate = 0.3), followed by a three-class Softmax output.

An additional preprocessing step specific to the neural network was applied: StandardScaler normalization of the two continuous features (Week and Shift) using parameters estimated exclusively from the training set, ensuring that the One-Hot Encoded columns (983 binary features) remained unmodified.

Training used the Adam optimizer (learning rate = 0.001), sparse categorical cross-entropy loss, balanced class weights, EarlyStopping (patience = 15 epochs monitoring validation loss), and ReduceLROnPlateau (patience = 7, factor = 0.5). The model converged in 23 epochs, recovering weights from epoch 8 (best validation loss). The final Macro F1 on the validation set was 0.3816 with a generalization gap of 0.3204 — below all tree-based configurations and with a higher gap than XGBoost and LightGBM under gap-penalized Optuna.

3.12. Final evaluation on the Held-out test set

The definitive performance of each model was assessed on the held-out test set (n = 965 records), which was not accessed at any point during model development, hyperparameter tuning, or validation. This strict temporal separation ensures that the reported metrics reflect genuine out-of-sample generalization under realistic deployment conditions

Table 7. Presents the Macro F1, Weighted F1, and the difference between validation and test performance for all four models

Model	Optimization	F1 Macro Val	F1 Macro Test	F1 Weighted Test	ΔVal→test
Random Forest	Manual Reg	0.3870	0.3748	0.4950	-0.0122
XGBoost	Optuna+Gap	0.4125	0.3829	0.4871	-0.0296
LightGBM	Optuna+Gap	0.4065	0.3568	0.4191	-0.0497
Neural Network	Dropout+L2	0.3816	0.3536	0.5512	-0.0280

ΔVal→Test represents the difference between validation and test Macro F1. Negative values indicate a slight drop in performance when generalizing to unseen data, which is a normal and expected behavior during model evaluation.

XGBoost with gap-penalized Optuna achieved the highest Macro F1 on the test set (0.3829), confirming its status as the best-performing model. Random Forest showed the smallest degradation (−0.0122. The Neural Network, while achieving the highest Weighted F1 (0.5512) due to strong Low-criticality detection, underperformed in Macro F1 (0.3536), reflecting its inability to classify Medium-criticality failures effectively.

Table 8. Presents the per – class classification report for XGBoost on the test set as the best model.

Class	Precision	Recall	F1 Score	Support	%
High (>44min)	0.290	0.321	0.304	159	16.5%
Low (</=23min)	0.690	0.534	0.602	629	65.2%
Medium (24-44 min)	0.192	0.328	0.242	177	18.3%
Macro Avg	0.391	0.394	0.383	965	100%
Weighted Avg	0.533	0.461	0.487	965	

Support: actual number of examples per class in the test set. %: percentage of the total examples represented by that support.

The per-class results reveal a consistent pattern across all models: Low-criticality failures (duration ≤ 23 min) are classified with precision (F1 = 0.602), while High-criticality failures (duration > 44 min) are detected with moderate recall at the cost of precision (F1 = 0.304), and Medium-criticality failures (24–44 min) remain the most difficult to classify (F1 = 0.242). This gradient of difficulty is structurally expected: Medium failures occupy an intermediate temporal range whose administrative records share characteristics with both adjacent classes

4. Discussion

The best model obtained XGBoost optimized with Optuna using a gap-penalized objective function achieved a Macro F1 score of 0.3829 on the test set. This study used exclusively administrative records from corrective work orders, where failure duration depends on factors not captured in the log: availability of spare parts at the time of failure, the assigned technician's profile, accumulated equipment wear, and shift-specific operating conditions. In this context, the obtained Macro F1 reflects the informational ceiling of the available data.

The three baseline models presented generalization gaps considered high. This finding is expected because the default parameters of scikit-learn are optimized to maximize training fit without complexity restrictions. Random Forest, with unlimited depth and a minimum of one record per leaf, achieved a training Macro that show clear evidence of memorization.

The regularization strategy via exhaustive search of complexity parameters significantly reduced the gap for all three models while simultaneously improving the validation Macro F1. This result confirms that the severe overfitting of the baseline models was not a consequence of the data relationships but of the absence of complexity constraints, and that regularization is a methodologically indispensable step prior to any hyperparameter optimization. The implementation of Optuna with an objective function that explicitly penalizes the gap proved to be the most effective strategy for jointly controlling performance and generalization

The decision to perform a chronological split of the dataset instead of a random split had direct implications for the validity of the results. A random split would have allowed the model to learn from failures occurring in October 2024 to predict failures from January 2024 an impossible condition in real production that generates artificially optimistic estimates. The implemented chronological split (70/15/15 by temporal position) replicated the real deployment scenario: the model learns from past history and predicts future events.

4.1 Limitations

The main limitations of the study are threefold. First, the exclusive reliance on administrative records without sensor data structurally limits the predictive content of the input variables. Integrating IoT signals of vibration, temperature, or electric current could substantially improve the Macro F1 by providing temporal patterns prior to failure that are not available in work order logs. Second, the twelve-month time horizon limits the ability to capture complete seasonal patterns and the effect of structural changes in the plant's operating conditions. Third, the presence of 65 machines with no failure history in training introduces noise in predictions for those specific assets.

References

1. [1] A. A. Sisode and M. Devare, "A Review on Machine Learning Techniques for Predictive Maintenance in Industry 4.0," *International Research Journal of Engineering and Technology (IRJET)*, vol. 9, no. 8, pp. 1169–1173, Aug. 2022.
2. [2] E. Samatas, S. Siatras, and I. Chatzigiannakis, "Predictive Maintenance: Bridging Artificial Intelligence and IoT," in *Proceedings of the 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Dalian, China, Jun. 2021, pp. 464–471.
3. [3] P. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, Jun. 2015.
4. [4] C. Campos-Olivares, D. Hernández-Gress, R. Rodríguez-Molina, and E. Olivares-Benitez, "A Systematic Mapping of the Advancing Use of Machine Learning Techniques for Predictive Maintenance," *DYNA*, vol. 91, no. 1, pp. 73–81, Jan. 2024.
5. [5] I. Hector and R. Panjanathan, "Predictive maintenance in Industry 4.0: A survey of planning models and machine learning techniques," *PeerJ Computer Science*, vol. 10, p. e2016, 2024, doi: 10.7717/peerj-cs.2016.
6. [6] B. van Oudenhoven, P. Van de Calseyde, R. Basten, and E. Demerouti, "Predictive maintenance for Industry 5.0: Behavioural inquiries from a work system perspective," *International Journal of Production Research*, vol. 61, no. 22, pp. 7846–7865, 2023, doi: 10.1080/00207543.2022.2154403.
7. [7] P. Karrupusamy, "Machine learning approach to predictive maintenance in manufacturing industry – A comparative study," *Journal of Soft Computing Paradigm* (JSCP), vol. 2, no. 4, pp. 246–255, 2020, doi: 10.36548/jscp.2020.4.006.
8. [8] D. C. Montgomery, Design and Analysis of Experiments, 9th ed. Wiley, 2017.
9. [9] J. Han, M. Kamber, y J. Pei, Data Mining: Concepts and Techniques, 3ra ed. Morgan Kaufmann, 2011
10. [10] T. Wireman, Developing Performance Indicators for Managing Maintenance. Morgan Kaufmann, 2004.
11. [11] B. W. Silverman, Density Estimation for Statistics and Data Analysis. Chapman and Hall, 1986.
12. [12] S. Sundaram and A. Zeid, "Technical language processing for Prognostics and Health Management: applying text similarity and topic modeling to maintenance work orders," Journal of Intelligent Manufacturing, Feb. 2024.
13. [13] S. Nakajima, Introduction to TPM: Total Productive Maintenance. Productivity Press, Cambridge, MA, 1988.
14. [14] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in Proc. 13th IJCAI, Chambéry, France, 1993, pp. 1022–1027.
15. [15] M. Sivakumar, S. Parthasarathy, and T. Padmapriya, "Trade-off between training and testing ratio in machine learning for medical image processing," PeerJ Computer Science, vol. 10, p. e2245, 2024. doi: 10.7717/peerj-cs.2245
16. [16] Y. Xu and R. Goodacre, "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning," Journal of Analysis and Testing, vol. 2, pp. 249–262, 2018. doi: 10.1007/s41664-018-0068-2
17. [17] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3rd ed. Melbourne, Australia
18. [18] H. R. Friesacher, E. Svensson, S. Winiwarter, L. Mervin, A. Arany, and O. Engkvist, "Temporal distribution shift in real-world pharmaceutical data: Implications for uncertainty quantification in QSAR models," *Artificial Intelligence in the Life Sciences*, vol. 8, p. 100132, 2025, doi: 10.1016/j.ailsci.2025.100132.
19. [19] J. Bergstra and Y. Bengio, "Random search for Hyper-Parameter Optimization," 2012. https://jmlr.org/beta/papers/v13/bergstra12a.html
20. [20] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, San Francisco, CA, 2016, pp. 785–794. doi: 10.1145/2939672.2939785
21. [21] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in Proc. 31st Conf. Neural Information Processing Systems (NIPS 2017), Long Beach, CA, 2017.
22. [22] R. Ballester, X. Arnal Clemente, C. Casacuberta, M. Madadi, C. A. Corneanu, and S. Escalera, "Predicting the generalization gap in neural networks using topological data analysis," *arXiv preprint arXiv:2203.12330v2*, 2023.
23. [23] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
24. [24] P. Probst, M. N. Wright, and A. L. Boulesteix, "Hyperparameters and tuning strategies for random forest," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 9, no. 3, e1301, 2019.
25. [25] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining, Anchorage, AK, 2019, pp. 2623–2631. doi: 10.1145/3292500.3330701