Machine Learning for Pattern Detection in Printhead Nozzle Logging

Nikola Prianikov, Marcin Pietrasik, Charalampos S. Kouzinopoulos
Department of Advanced Computing Sciences
Maastricht University
Maastricht, The Netherlands

Evelyne Janssen
Quality Processes and Validation Department
Canon Production Printing
Venlo, The Netherlands

Abstract—Correct identification of failure mechanisms is essential for manufacturers to ensure the quality of their products. Certain failures of printheads developed by Canon Production Printing can be identified from the behavior of individual nozzles, the states of which are constantly recorded and can form distinct patterns in terms of the number of failed nozzles over time, and in space in the nozzle grid. In our work, we investigate the problem of printhead failure classification based on a multifaceted dataset of nozzle logging and propose a Machine Learning classification approach for this problem. We follow the feature-based framework of time-series classification, where a set of time-based and spatial features was selected with the guidance of domain experts. Several traditional ML classifiers were evaluated, and the One-vs-Rest Random Forest was found to have the best performance. The proposed model outperformed an in-house rule-based baseline in terms of a weighted F1 score for several failure mechanisms.

Index Terms—feature engineering, time-series classification, corrective maintenance, printing, nozzle log

I. INTRODUCTION

Identifying failure mechanisms is a critical part of industrial corrective maintenance for manufacturers to ensure the quality of their products [1]. Many manufactured systems are made up of smaller components that can fail over time, eventually causing entire system breakdown. By monitoring the condition of these individual parts, unique patterns of wear and tear can be uncovered, which can then be linked to potential causes of failure. An example of this is the inkjet printheads produced by Canon Production Printing (CPP) for high-volume printing. The performance of these printheads depends heavily on the individual drop-forming nozzles, whose failure logging is constantly recorded. By analyzing nozzle failure development patterns in time and space over the nozzle grid from the logging data, failure mechanisms can be identified. The CPP team has observed several distinct patterns of nozzle failures that correspond to the failure mechanism of a complete printhead. For instance, one pattern may show a scattered combination of nozzle blockages, while another could reveal a large number of neighboring nozzles failing simultaneously due to an electrical issue.

To classify failures, manufacturers typically use two main approaches: rule-based methods and algorithmic-based models. Rule-based methods rely on organizing domain knowledge into a hierarchy of rules which produce classifications, while algorithmic approaches are data-driven, often leveraging Machine Learning (ML) techniques [2]. Domain experts at CPP have developed an in-house rule-based classification model to identify failure mechanisms from the terminal nozzle log records based on expert knowledge and historical observations. Thus, when the classifier makes incorrect predictions for new data points, the rules need to be manually adjusted, or the predictions must be overwritten by hand. In contrast, ML models can automatically learn relationships between input data and output labels, allowing them to generalize better to unseen scenarios [1].

In this paper, we address a unique industrial challenge: classifying failure mechanisms of printheads at their End-of-Life (EoL) stage using a multifaceted nozzle logging dataset. We propose an ML-based failure classification framework that integrates both time-series and spatial aspects of the data. Our approach builds on feature-based time-series classification methods, adapted to this specific application in collaboration with domain experts. Following extraction of useful features, an optimal classifier was built by fine-tuning and evaluating several traditional ML models and their One-vs-Rest (OVR) variants. Our results have shown that the OVR Random Forest (RF) model performed best, achieving an average weighted F1 score of 0.93 and reaching the human-level performance of the in-house rule-based baseline model (hereafter, the baseline).

II. RELATED WORK

Industrial time-series classification datasets often focus on general manufacturing or industry-specific cases, such as automotive or healthcare. These datasets typically consist of conventional sensor readings capturing variables like vibration, temperature, and motion [3]. A more complex case that happens in industry is in the time-series of historic log records such as sequences of error codes [4].

In contrast, our problem resembles the identification of system failures based on the failure developments of its components. This results in a time-series representing the count information of how many component failures have been logged for the whole system. Thus, this type of data is sometimes formally referred to as the *count* time-series [5] to highlight the difference between classical cases such as sensor-based. This problem is not well studied in the ML literature, but is mainly addressed with comprehensive statistical frameworks such as FMEDA [6], which relies on the estimates of failure rates of individual components. Moreover, existing research on CPP

printheads has primarily addressed lifespan estimation [7], while classification of printhead failures from nozzle logging has not been studied yet.

Time-Series Classification (TSC) has become a critical tool in modern industrial applications. It encompasses a variety of methods, among which feature-based algorithms are highlighted as an effective approach [8]. Over the years, a wide range of feature-based techniques has been developed, from Shapelet-based like Fast Shapelets [9] to the hybrid state-of-the-art approaches such as HIVE-COTE 2.0 [10]. Another notable family of feature-based techniques focuses on statistical feature engineering. Recent frameworks like Tsfresh [11] and TSFEL [12] automate feature mining and selection using statistical tests, and they are particularly relevant for multi-variate time-series problems. However, these methods can be computationally intensive, as they often calculate a large number of features, many of which may be irrelevant to the specific problem [13].

III. METHODOLOGY

A. Nozzle logging and failure mechanisms of printers

Typically, when printheads fail, it is possible to establish the associated failure mechanism based on the physical state or the error logging of that printhead. When several printheads fail for unknown reasons but exhibit similar nozzle log activity, it can be assumed they have the same failure mechanism. A printhead's nozzle log consists of several components among which one can find the status of individual nozzles according to Nozzle Failure Classification (NFC), which indicates that the nozzle does not jet the ink properly. If a nozzle is not behaving correctly, then it is assigned one of the five status labels *Nozzle Failure [1-5]* (*NF[1-5]*). The labels are anonymized to preserve the privacy of the internal terminology of CPP.

Formally, a nozzle log of a printhead is a time series $\mathcal{T}^i$ which is defined as a sequence of log records:

$$\mathcal{T}^i = \{X^{(1)}, X^{(2)}, \ldots, X^{(n)}\}, \tag{1}$$

Where n is the total number of log records and i indexes the specific printhead. Each time series $\mathcal{T}^i$ of a removed printhead is associated with a corresponding failure mechanism label y^i. Each log record, $X^{(t)}$, captures the state of the printhead's 512 nozzles at a time-step t. The nozzles are arranged in four rows of length 128 resulting in a matrix with dimensions 4×128. Each matrix entry indicates the state of a nozzle, where $NF[1-5]$ and $\emptyset$ denote possible NFC states:

$$X^{(t)} \in \{NF1, NF2, NF3, NF4, NF5, \emptyset\}^{4 \times 128},$$
$$\forall t \in \{1, 2, \ldots, n\} \tag{2}$$

Alternatively, each log record $X^{(t)}$ can be viewed as a 5-channel image of size $4 \times 128 \times 5$:

$$X^{(t)} \in \{0, 1\}^{4 \times 128 \times 5}, \quad \forall t \in \{1, 2, \ldots, n\}. \tag{3}$$

In this representation, 4 and 128 denote spatial dimensions of an image, corresponding to nozzle layout, while 5 represents the number of channels. Each channel is a binary grid indicating whether a nozzle is in one of the five possible statuses $NF[1-5]$ at a time-step t. This representation allows to view nozzle log records as images, as illustrated in Fig. 1, enabling visual identification of patterns. The grid plot on the left of Fig. 1 represents the full terminal record. The time-series on the right illustrates developments of the number of NFCs over time aligned over the last 100 print jobs of a printhead. Each timestamp corresponds to the count of NFCs in the first nozzle log record of a print job while the color indicates the NFC type.

B. Dataset

The dataset used in this case study consists of 411 printheads, which were removed from printers in the field due to failures in the nozzle log. The printheads in the dataset are categorized into six failure mechanisms of *Pattern[1-5]* and *Pattern 1&2*. The class distribution in the dataset is heavily imbalanced, as the most populated class *Pattern 1* accounts for 121 samples, while the least populated class *Pattern 5* has only 23. Classes of *Pattern 1* through *Pattern 5* correspond to the failure mechanisms that can be clearly identified from the patterns in the nozzle log data. Cases that do not fit these patterns are grouped under a general *Other* category, which includes printheads with sparse logs or atypical behaviors that lack sufficient data to constitute a separate class. The dataset also contains six printheads that exhibit characteristics of both *Pattern 1* and *Pattern 2*. While these instances could be treated as edge cases, we adopt a multi-label classification approach and include them as valid samples for both respective classes.

Class labels were assigned by the domain experts at CPP through a multi-stage process. Initially, failure mode predictions were generated using the in-house baseline model. Predicted labels were then verified by the domain experts, which involved a manual inspection of the terminal nozzle log record, usage history of each printhead and time-series information of NFC developments. Eventually, any incorrectly predicted labels were manually reviewed to ensure accuracy.

C. Model design

1) Data extraction and representation: A significant portion of the raw nozzle log may be superfluous, since new records are added with a high frequency, while NFC state is unlikely to change that rapidly. Additionally, raw nozzle log data poses challenges for storage and increases the time and space complexity of classification algorithms. Thus, when extracting raw nozzle log from the database (denoted as *DB* in Fig. 2), the data is sampled such that only the first record of every print job is considered. To obtain count time-series data from the nozzle log, the number of failed nozzles per NFC type is taken from each time-step, transforming $\mathcal{T}^i$ into $\mathcal{T}'$. This results in a dataset of multi-variate time-series where each channel contains the count information of a specific nozzle failure type, as illustrated in Fig. 1.
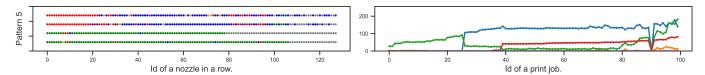
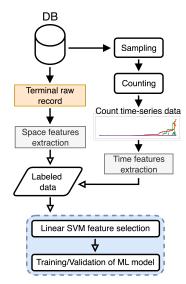Fig. 1: Example of a pattern in the nozzle log.

Fig. 2: Overview of the modeling framework.

Next, we use Tsfresh Python package [11] as it allows the extraction of various features from time-series. Out of the 60 time-based feature functions provided by Tsfresh, only 11 were selected based on a combination of domain expertise and empirical evaluation. The chosen features capture key characteristics such as linear trends, autocorrelation, complexity, and other relevant statistical properties.

In addition to the Tsfresh-based features, seven custom functions were introduced to capture complementary aspects of the time series. These include the first and second derivatives, the final value of the time series, and the maximum difference between consecutive samples. The choice of these features was strongly guided by CPP domain experts. Derivative-based features enhance the model's ability to distinguish subtle temporal variations, while the final value is critical as it plays a central role in the decision-making logic of the baseline model.

Finally, space feature functions were added to extract properties from the terminal nozzle log state of printheads. Two such functions were added: the average position of failed nozzles per NFC and the number of consecutive NF4s from the edge of a grid. Overall, 20 time-based feature functions were parameterized and applied to the nozzle log, resulting in 430 numeric features per instance of a failed printhead.

2) Development of an optimal classifier: Initially, an extensive set of ML models available through the *scikit-learn* library was evaluated with the *pycaret* package. Based on evaluation

results, it was decided to proceed with the RF, Logistic Regression (LR), Extremely Randomized Trees (ET), Decision Tree (DT) and K-Nearest Neighbors (KNN) methods, as well as with Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel. For each of these methods, a pipeline of imputation, standard scaling and feature selection was used. To discard irrelevant features and improve generalization, a model-based feature selection procedure was employed using a linear kernel SVM with a tight regularization coefficient of 0.01. These components of the modeling framework are represented by the blue rectangle in Fig. 2.

Finally, the problem was transformed into a multi-label classification setting, as a handful of samples exhibited both *Pattern 1* and *Pattern 2* failure mechanisms at the end of their lifetime. To allow predicting multiple labels, classifiers were adapted to the OVR framework. Specifically, a separate binary classifier is trained for each class against all others, and the final decision is obtained by combining their outputs.

IV. EVALUATION

A. Evaluation procedure and metrics

Due to the imbalanced dataset and a limited number of samples for several classes, all of the comparisons between the models and training settings were performed using the leave-one-out cross validation (LOOCV) framework. After the split-wise predictions were obtained, the precision, recall and F1 scores were subsequently calculated. Precision can be interpreted as the ratio of correctly predicted samples over the total number of predictions, while recall corresponds to the ratio of true samples that were accurately classified. The F1 score aggregates these two values with a harmonic mean, providing an overall outlook on the performance of the model. For the evaluation of multi-labeled classifiers, it is necessary to properly average out the scores according to one of the strategies, namely micro-average, macro-average, and weighted-average. We used weighted-average as this type of averaging is particularly suitable for classification problems with class imbalance as it ensures a fairer evaluation.

B. Results

1) Selection of the highest performing ML model: To identify the best performing ML model, the hyperparameters for each classifier were tuned using a 10-fold cross validation procedure. Next, the LOOCV performance was assessed for each of the models with the weighted scores being summarised in Table I. Since the primary focus of this experiment is on discriminating between the five well-defined failure patterns, the weighted averages were computed by excluding the *Other*

TABLE I: Evaluation results of the tuned ML models.

Model	Prec.	Rec.	F1
RF	0.9318	0.9513	0.9410
LR	0.9201	0.9294	0.9242
KNN	0.8366	0.8394	0.8369
DT	0.8077	0.8881	0.8428
ET	0.9355	0.9440	0.9392
SVM	0.9170	0.9367	0.9266

class from the evaluation. As it can be observed from the Table I the Random Forest consistently outperforms other classifiers in terms of the F1 score and recall. Based on these findings, a Random Forest model with 50 trees, a maximum depth of 20 and the Gini impurity criterion was selected for use in the subsequent round of experiments.

2) Evaluation of an optimal ML classifier against the rule-based baseline: Following the identification of the optimal classification model in the initial experiment, it is now possible to compare the performance of the developed classifier to the baseline on the full dataset. While direct comparisons between rule-based and ML methods are typically inappropriate due to their fundamentally different algorithmic approaches, such a comparison is justified in this case. The rule-based baseline, developed by domain experts, establishes a higher validation boundary than a random classifier. Furthermore, the absence of ML baselines for this specific problem makes the rule-based approach a reasonable benchmark against which our model's performance can be evaluated.

The evaluation scores summarizing the performance of the baseline and ML model are presented in Table II. From the results, it is evident that the ML model clearly outperforms the baseline in predicting *Pattern 2*, *Pattern 4*, and *Pattern 5* as reflected in terms of the F1 scores. These findings are further supported by the confusion matrices in Fig. 3. Note that because OVR classifiers may output multiple predictions, the raw sums can exceed the actual class support values. However, this does not undermine their usefulness for comparing class-wise prediction tendencies. At the same time, the baseline demonstrates strong superiority in predicting *Pattern 3* failures and performs slightly better for *Pattern 1*, while the ML model almost reaches its recall at 0.96. For *Pattern 2*, the ML model has a higher precision of 0.95, however the baseline outperforms its recall with an almost perfect score of 0.99. Finally, in terms of overall misclassifications, the ML model demonstrates improved performance, with 31 incorrect predictions compared to 39 from the baseline.

3) Feature importance in class predictions: The use of a Random Forest model as the primary classification approach allows for clear interpretation of feature importance through the Gini index. By analyzing the top 10 features for each class, we gained insights into the model's decision-making process. Notably, custom features created with domain knowledge, such as the number of consecutive NF4s, maximum differences and derivatives, were among the most relevant for predicting

TABLE II: Classification report of the rule-based baseline and OVR Random Forest model for the five common output labels, transformed into multi-label problem.

	Model	Prec.	Rec.	F1	Support
Pattern 1	Baseline	0.93	0.98	0.95	127
	OVR RF	0.90	0.96	0.93	
Pattern 2	Baseline	0.80	0.99	0.89	75
	OVR RF	0.95	0.93	0.94	
Pattern 3	Baseline	1.0	1.0	1.0	30
	OVR RF	0.80	0.93	0.86	
Pattern 4	Baseline	0.76	0.73	0.75	26
	OVR RF	1.00	0.85	0.92	
Pattern 5	Baseline	0.92	0.52	0.67	23
	OVR RF	0.95	0.87	0.91	
Other	Baseline	0.97	1.00	0.99	136
	OVR RF	0.96	0.93	0.94	
Weighted average	Baseline	0.91	0.95	0.93	417
	OVR RF	0.93	0.93	0.93	

Pattern 2. Similarly, for *Pattern 1*, maximum differences and linear trend features derived from domain expertise were given the highest importance scores. In addition, domain-informed features also played a significant role for the correct classification of *Pattern 4*.

C. Discussion

The results from the first experiment indicate that the Random Forest model demonstrates superiority in terms of F1 score and recall, likely due to its ability to learn non-linear decision boundaries through ensemble learning. The LR and SVM models perform similarly well, possibly benefiting from the well-designed set of features. In contrast, DT and KNN exhibit the worst performance, likely due to overfitting and poor generalization on the small dataset.

In the main experiment of this study, the shortcomings of the rule-based model were clearly summarized in the classification report for its predictions in Table II and in the confusion matrices on Fig. 3. The performance drawbacks mainly stem from the high number of False Positives (FPs) for *Pattern 2* and the high number of False Negatives (FNs) for *Pattern 5*. This indicates that the baseline algorithm has a simple criterion for assigning *Pattern 2* and struggles to detect many instances of *Pattern 5*. Additionally, a relatively high number of FNs is present for *Pattern 4*.

It was demonstrated that an ML classifier outperformed the baseline in terms of the number of misclassified samples and achieved a comparable performance in terms of the weighted F1 score. Key limitations include struggles with the class *Other*, as the 10 instances of this class were misclassified. Additional confusions occurred between *Pattern 1* and *Pattern 2* classes, which even domain experts find hard to distinguish.

Baseline

	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Other
Pattern 1	124	12	0	0	1	0
Pattern 2	2	74	0	0	0	0
Pattern 3	0	0	30	0	0	0
Pattern 4	6	4	0	19	0	1
Pattern 5	2	2	0	6	12	3
Other	0	0	0	0	0	136

OVR RF

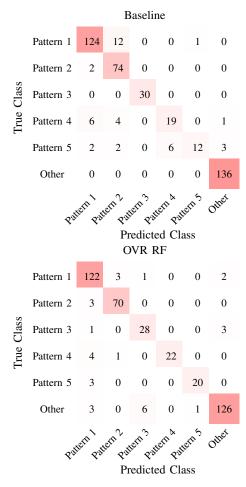|	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Other
Pattern 1	122	3	1	0	0	2
Pattern 2	3	70	0	0	0	0
Pattern 3	1	0	28	0	0	3
Pattern 4	4	1	0	22	0	0
Pattern 5	3	0	0	0	20	0
Other	3	0	6	0	1	126

Fig. 3: Confusion matrices of prediction results of the rule-based baseline model and proposed ML model across common output labels.

Errors also arose for the *Pattern 3* samples where nozzle log was mostly empty and only several failure records were present at the terminal stage of the printhead's lifespan, suggesting that time-based feature functions could not adequately capture such edge cases. This highlights the need for further refinement of features, though some cases remain challenging to discriminate even for domain experts.

The feature importance scores retrieved for the Random Forest model provided a valuable insight that most of the custom features designed with the domain knowledge were important for predicting the *Pattern 1*, *Pattern 2* and *Pattern 4* classes.

V. CONCLUSION

This study has addressed the problem of classification of failure mechanisms of CPP's printheads caused by errors in nozzle logging highlighting the importance of synergy between data-driven approaches and domain knowledge in industrial corrective maintenance. An ML classifier was successfully developed and demonstrated to outperform the rule-based baseline for specific failure patterns, reaching an overall F1

score of 0.93. It was determined that OVR RF performed best across evaluated models and exceeded the predictive performance of rule-based baseline with respect to three out of five classes. It was determined that the problem of multi-variate count time-series classification in the context of limited data can be effectively addressed by transforming each individual time-series into a fixed-length feature vector using time-based feature functions. The proposed classification framework and feature engineering approach are relevant for analyzing similar systems of multiple small parts prone to failure or degradation over time.

REFERENCES

[1] Mohamed Er-Ratby, Abdessamad Kobi, Youssef Sadraoui, and Moulay Saddik Kadiri. The Impact of Predictive Maintenance on the Performance of Industrial Enterprises. *SN Computer Science*, 6(1):73, January 2025.

[2] Guenter Roehrich and Davide Raffaele. Predictive maintenance: advanced fault classification. *International Journal of System Assurance Engineering and Management*, December 2024.

[3] Mojtaba A. Farahani, M.R. McCormick, Robert Gianinny, Frank Hudacheck, Ramy Harik, Zhichao Liu, and Thorsten Wuest. Time-series pattern recognition in Smart Manufacturing Systems: A literature review and ontology. *Journal of Manufacturing Systems*, 69:208–241, August 2023.

[4] Antoine Guillaume, Christel Vrain, and Elloumi Wael. Predictive maintenance on event logs: Application on an ATM fleet, November 2021.

[5] Richard A. Davis, Konstantinos Fokianos, Scott H. Holan, Harry Joe, James Livsey, Robert Lund, Vladas Pipiras, and Nalini Ravishanker. Count Time Series: A Methodological Review. *Journal of the American Statistical Association*, 116(535):1533–1547, July 2021.

[6] John C Grebe and William M Goble. Fmeda–accurate product failure metrics. *FMEDA Development Paper, Rev*, 1, 2007.

[7] Dan Parii, Evelyne Janssen, Guangzhi Tang, Charalampos Kouzinopoulos, and Marcin Pietrasik. Predicting the lifespan of industrial printheads with survival analysis. In *2025 IEEE 8th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 1–6, 2025.

[8] Mojtaba A. Farahani, M.R. McCormick, Ramy Harik, and Thorsten Wuest. Time-series classification in smart manufacturing systems: An experimental evaluation of state-of-the-art machine learning algorithms. *Robotics and Computer-Integrated Manufacturing*, 91:102839, February 2025.

[9] Aaron Bostrom and Anthony Bagnall. Binary shapelet transform for multiclass time series classification. In *Big Data Analytics and Knowledge Discovery: 17th International Conference, DaWaK 2015, Valencia, Spain, September 1-4, 2015, Proceedings 17*, pages 257–269. Springer, 2015.

[10] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. HIVE-COTE 2.0: A new meta ensemble for time series classification. *Machine Learning*, 110(11-12):3211–3243, December 2021.

[11] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, 307:72–77, September 2018.

[12] Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[13] Mohamed-Ali Tnani, Paul Subarnaduti, and Klaus Diepold. Efficient Feature Learning Approach for Raw Industrial Vibration Data Using Two-Stage Learning Framework. *Sensors*, 22(13):4813, June 2022.