A hybrid physical-machine learning framework for predictive inspection scheduling in integrated BIM-IoT environments

Tetyana Honcharenko† and Olga Solovei*,†

Kyiv National University of Construction and Architecture, Air Force Avenue 31, 03037 Kyiv, Ukraine

Abstract

The construction industry's reliance on fixed, calendar-based inspections is inefficient and often misses gradual infrastructure degradation, causing costly reactive repairs. This paper presents a six-step framework combining a contextual graph model, real-time IoT data, and a hybrid physics–machine learning pipeline for predictive inspection scheduling. A physics-based degradation model simulates corrosion, fatigue, and stress to create a synthetic dataset for training. A Random Forest model achieved accuracy in detecting at-risk states and a forecasting algorithm predicting inspection needs. Applied to a static plan, the framework detects new high-risk periods, validates or adjusts scheduled inspections, and defers low-risk ones. By updating maintenance plans with data-driven insights, it enables proactive, condition-based asset management that improves safety and reduces costs.

Keywords

Predictive Maintenance, Internet of Things, Building Information Modeling, Machine Learning, Graph Database.

1. Introduction

Construction industry standards: DSTU 29481-1:2022, DSTU 29481-2:2023 specifies fourteen types of data for construction projects: Project, BIM Model, BIM Element, Construction Site, GIS Layer, Document, Multimedia, Telemetry, Event/Incident, Task, Financial Transaction, Resource/Contractor, Inspection (Operational), Provenance/Lineage.

Modern construction standards, such as DSTU 9243.4:2023, mandate robust asset monitoring throughout a project's lifecycle. However, conventional monitoring often fails to detect the slow, insidious degradation of critical components. Consequently, the industry predominantly relies on a reactive, calendar-based inspection process (Figure 1), which is often inefficient and poorly timed [1]. A compelling example is the slow water ingress in a concrete foundation, which can lead to rebar corrosion. This degradation process may proceed undetected for years, and by the time a scheduled inspection occurs, the structural damage may be severe and prohibitively expensive to repair. This gap highlights the urgent need to transition from rigid, calendar-based schedules to intelligent, predictive maintenance methodologies.

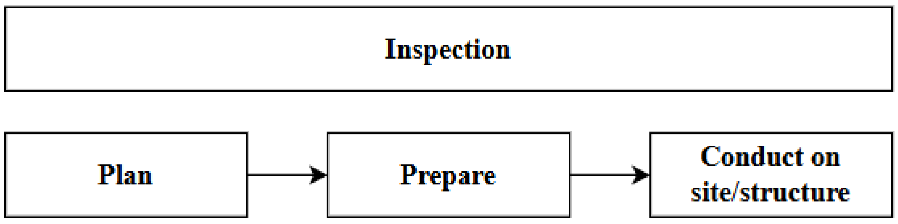


Figure 1: A calendar-based Inspection process.

AIT&AIS'2025: International Scientific Workshop on Applied Information Technologies and Artificial Intelligence Systems, December 18–19 2025, Chernivtsi, Ukraine

1* Corresponding author.

† These authors contributed equally.

iust511@ukr.net (T. Honcharenko); solovey.ol@knuba.edu.ua (O. Solovei)

0000-0003-2577-6916 (T. Honcharenko); 0000-0001-8774-7243 (O. Solovei)

The convergence of Digital Twin, the Internet of Things (IoT), and machine learning (ML) offers a solution to address this challenge. Yet, as identified in a comprehensive literature review [2], the field still requires more practical, implementable frameworks and advanced algorithms to realize its full potential.

This paper directly answers that call. We propose a novel framework for predictive inspection scheduling designed to preempt costly failures by detecting the degradation. Our methodology leverages a hybrid approach: it integrates real-time data from embedded sensors with the static, contextual knowledge of an asset, such as material specifications and structural load models. A machine learning model trained on this combined data forecasts the future need for an inspection. Upon a positive prediction, our framework automatically generates a targeted inspection task, transforming the maintenance process from a reactive, time-based cycle into a proactive, data-driven, and condition-based workflow.

2. Research Objectives

The goal of this research is to design and validate a framework that facilitates a shift from calendar-based inspection schedules to a predictive maintenance strategy for civil infrastructure.

To achieve this objective, this study will resolve the tasks:

1. Develop a physical degradation model of a specific BIM element by translating sensor data into physical metrics.
2. Design an algorithm for forecasting Inspection Timeline.
3. Define a method to train and evaluate machine learning classification model.

The scope of this initial study is focused on demonstrating the methodology's viability by developing and applying the physical degradation model to a single, asset type: a structural steel beam.

3. Literature Review

The research in [3] proposes a Digital Twin framework for railway turnouts, with a core contribution centered on Explainable AI (XAI). By modeling the system with a Bayesian Network structured by expert engineering knowledge, their approach delivers not just predictions but also probabilistic explanations for impending failures. While this 'white-box' model is highly valuable, its reliance on pre-defined causal structures can be challenging to scale across the diverse and often less-understood failure modes of heterogeneous building assets. Our framework complements this by proposing a hybrid approach that leverages both physics-based models and data-driven machine learning, offering adaptability where a single, explicit causal model may be insufficient.

The framework developed in [4] for wind turbines focuses on real-time prognostics and active control. Their Digital Twin not only forecasts failures but also tests control strategies to bring deviating parameters back within operational thresholds, creating a self-regulating system. Our perspective is that this self-regulation addresses the symptom, it cannot replace physical intervention to resolve the root cause of degradation. Therefore, we treat the frequency and magnitude of such control actions as a primary input signal for our inspection prediction model.

The work in [5] addresses the pre-deployment problem of optimal sensor placement. Their results prove that a well-designed sensor network, derived from physics-based simulations, maximizes the signal-to-noise ratio for damage detection. Our research is complementary and focuses on the subsequent operational phase. Assuming such a well-placed network is providing data, our framework answers the question of how to integrate this live sensor data with the rich contextual information from BIM models to drive a predictive workflow. While simulation-based design is essential, it is insufficient on its own, as it operates under idealized conditions and cannot account for the operational complexities our framework is designed to handle.

The study in [6] successfully validates a real-time infrastructure monitoring system for a "Noise Barrier Tunnel", integrating a high-resolution BIM model with live IoT data. The system proved effective through a dual-layer anomaly detection strategy combining real-time hardware alerts with offline analysis of subtle irregularities. A key limitation, stated by the authors, is that these advanced data-driven analytics are not part of the real-time system. Our research addresses this exact gap by automating and integrating the predictive analytics layer into the operational workflow. We achieve this by adding a machine learning classifier to forecast inspection needs in real-time, transforming the system from a reactive monitoring platform into a proactive one.

The research in [7] presents a comprehensive, six-layer data-driven Digital Twin framework designed to improve Facility Management for tunnels. A limitation of this otherwise holistic study lies in its treatment of the predictive analytics layer as a generic "data-driven" module, which may lack the specificity and interpretability required for the diverse and physically distinct assets found in a BIM model. Our research addresses this limitation by proposing a hybrid physical-ML pipeline. Instead of a single black-box model, we first use domain-specific engineering principles to compute explicit degradation metrics, which then serve as robust features for a data-driven classification model. This provides a more transparent, adaptable, and physically grounded approach to prediction.

4. Materials and methods

4.1. A design of framework to enable predictive inspections for construction project asset

This paper proposes six-process framework designed to enable predictive inspections for building assets by integrating heterogeneous data sources through a hybrid physical-machine learning pipeline (Figure 2).

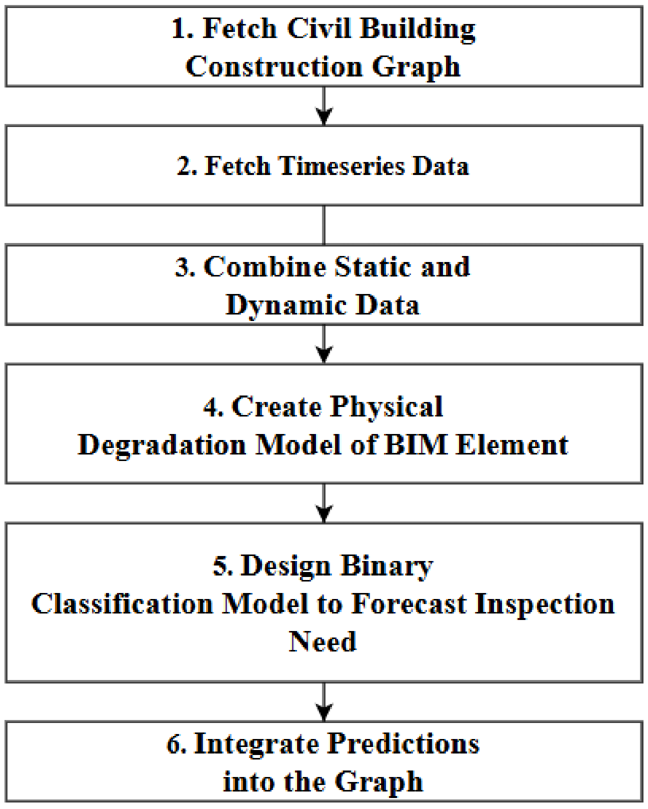

1. Fetch Civil Building Construction Graph
2. Fetch Timeseries Data
3. Combine Static and Dynamic Data
4. Create Physical Degradation Model of BIM Element
5. Design Binary Classification Model to Forecast Inspection Need
6. Integrate Predictions into the Graph

Figure 2: Six-process framework designed to enable predictive inspections for construction project assets.

At the core of our framework is a formal representation of the construction project as a heterogeneous property graph, denoted as $G = (V, E)$. Here, V represents the set of all project object types $O = \{o_1, \ldots, o_n\}$, where n is the number of object types, and E represents the multi-relational connections between them. This contextual graph is managed within a native graph database to ensure high-performance traversal and querying, in line with modern data management principles.

Since each object $o \in O$ is defined by different data types, the database for each must be selected as a "best-in-class" solution for that specific data type. For example, real-time time-series data from IoT sensors is best stored in a specialized database like InfluxDB [8].

Therefore, the framework includes processes to collect data about objects $o \in O$ from their respective databases and storage systems. The process "Fetch Civil Building Construction Graph" retrieves the contextual graph G. The "Fetch Time-Series Data" process retrieves sensor measurements for specific BIM elements over a defined time window. In the "Combine Static and Dynamic Data" process, static metadata from the project graph G is merged with dynamic sensor measurements to form a dataset suitable for training a machine learning classification model. The "Create Physical Degradation Model of BIM Element" process defines functions to compute degradation metrics.

The "Design Binary Classification Model to Forecast Inspection Need" process involves designing and building a machine learning model to predict whether an inspection is required within a specified future time horizon. The "Integrate Predictions into the Graph" process dynamically updates the graph model G with inspection predictions by creating new nodes and relationships to represent forecasted risks and inspection schedules.

4.2. Beam element Physical degradation model

Corrosion reduction factor [9]:

$$CF = 1.0 + \frac{D}{3650} \cdot \frac{H}{100} \cdot k_c, \tag{1}$$

where D – days in service, H – humidity, $k_c \in [0.05, 0.15]$ corrosion effect range.

The section modulus of a rectangular beam, adjusted for corrosion:

$$S = \frac{bh^2}{6 \cdot CF}, \tag{2}$$

where b – beam width, m; h – beam height, m.

Maximum bending moment for a centrally loaded beam:

$$M = \frac{PL}{4}, \tag{3}$$

where L – beam span length, m; P – weight of a single central load, kN.

Bending stress is calculated as the maximum bending moment divided by the section modulus:

$$\sigma = \frac{M}{S}. \tag{4}$$

Temperature-adjusted Yield Strength σy based on temperature T, assuming a reduction of approximately 0.02% per °C above 20°C [10]:

$$\sigma_{y,eff} = \sigma_y \cdot (1 - 0.0002 \cdot (T - 20)). \tag{5}$$

Safety factor is calculated by dividing the effective yield strength by the bending stress:

$$SF = \frac{\sigma_{y,eff}}{\sigma}. \tag{6}$$

Initial Stress Intensity Factor to measure the intensity of stress near a crack tip [11]:

$$K_I = \sigma \sqrt{\pi \alpha_0}, \tag{7}$$

where α_0 is an initial crack length, m.

Fatigue crack propagation over time in materials under cyclic loading, using Paris Law [12]:

$$\frac{d\alpha}{dN} = C(K_I)^m, \tag{8}$$

where $d\alpha/dN$ is a crack growth rate; c, m are material-dependent constants; N – a total number of loading cycles over the time period, calculated as:

$$N_{total} = n_c \cdot D, \tag{9}$$

where n_c – number of cycles loadings the component per day, D – number of days the component has been in service.

Total crack length after N cycles starting from initial crack length α_0 [13]:

$$\alpha_{final} = \alpha_0 + \frac{d\alpha}{dN} \cdot N. \tag{10}$$

A final stress Intensity Factor at the final crack length α_{final}:

$$K_{I,final} = \sigma \sqrt{\pi \alpha_{final}}. \tag{11}$$

The binary rule to identify when inspection is needed consists of 3 conditions, which are joined by logical "or" operations: a final crack length α_{final} exceeds 15 mm; safety factor SF is less than 2.0 indicating that the material is closed to failure under the given conditions; the final stress intensity factor $K_{I,final}$ exceeds 90% of the material's fracture toughness 90% of the material's fracture toughness K_{IC} [14]:

$$Inspection = \begin{cases} 1, \alpha_{final} > 0.015 \vee SV < 2.0 \vee K_{I,final} \geq 0.9 K_{IC} \\ 0, otherwise \end{cases}. \tag{12}$$

4.3. Algorithm for Forecasting Inspections Timeline

To identify when a beam requires inspection, this paper proposes an algorithm that integrates physical degradation modeling with predictive machine learning techniques. For the defined forecast period the algorithm identifies the number of periods denoted as n_{steps}, and for each period calculates the future date, denoted as t_{future}. It updates the beam's parameters using a time-dependent degradation model, denoted as X_{future}, using function $f(\cdot)$ which performs calculations according to equations (1–12). A trained binary classification model M, is used to predict probability of inspection needs at time t_{future}. The algorithm's output is a dataframe with forecasted beams inspection timeline, denoted as *df*.

Algorithm 1. Algorithm for forecasting Beam Inspection Timeline:

```
1. Input: A trained binary classification model $M$, dataset $D$ at moment $t = 0$, total days to
   forecast $T_{forecast}$, time interval between forecast steps $T_{interval}$.
2. Output: dataframe with forecasted beams inspection timeline df.
3. Initialization: $T_{current} \longleftarrow$ SystemDate(); $n_{steps} = T_{forecast} / T_{interval}$; timeline $\longleftarrow$ []
4. FOR step in range($n_{steps}$):
   1. $\Delta t$ = step × $T_{interval}$
   2. $t_{future} = t_{current} + \Delta t$.
   3. $X_{future}$= f ($t_{future}$)
   4. $X'_{future}$ = StandardScaler($X_{future}$)
   5. $P_{inspection}$ = M.predict($X'_{future}$)
   6. timeline.append([$t_{future}$, $\Delta t$, $P_{inspection}$, $X_{future}$]
   EndFor
   7. Return $df$ = DataFrame(timeline)
```

4.4. Method to train and evaluate machine learning classification algorithms

Random Forest Classifier is trained with 200 trees, a learning rate is set to 0.1 and a maximum tree depth limited to 15. Additionally, 10 samples are required to form a node. Gini Impurity is used as the splitting criteria:

$$G(t)=1-\sum_{i=1}^{k}(p_i)^2, \tag{13}$$

where p_i is the proportion of samples belonging to class i at node t.

The XGBoost classifier is trained with 300 trees, a learning rate of 0.1, and a maximum tree depth limited to 7. The objective function to be minimized is Binary Log Loss:

$$L=-\frac{1}{N}\sum_{i=1}^{N}[y_i\log(p_i)+(1-y_i)\log(1-p_i)]. \tag{14}$$

The hyperparameters for Random Forest Classifier and XGBoost were selected based on established best practices and findings from preliminary experiments. A standard scaling method is used to normalize feature values with the same scale.

The performance of each model is evaluated using the 5-fold cross-validation method and includes the following metrics:

The proportion of correctly classified samples among all samples:

$$Accuracy=\frac{TP+TN}{TP+FP+TN+FN}. \tag{15}$$

The harmonic mean of precision and recall, balancing the tradeoff between false positives and false negatives:

$$F_1=\frac{2\cdot Precision\cdot Recall}{Precision+Recall}. \tag{16}$$

Additionally, a comprehensive visualization of where errors occur in "No Inspection Required" vs. "Inspection Required" predictions is provided through a confusion matrix.

5. Experiments Preparation

5.1. Proposed Operational Data Integration Framework

To enable predictive inspections for construction project assets, we propose an operational framework that integrates data from multiple specialized databases.

A comprehensive civil construction project model, represented as a graph $G = (V, E)$, is stored in a Neo4j database (see Figure 3). The process of identifying data for a specific asset—in the scope of this paper, a construction beam—begins by querying this graph. A Cypher query, such as MATCH (s:Sensor)-[:ATTACHED_TO]->(e:Element {elementID: $target_id}) RETURN s.deviceID, is used to traverse the graph from a target :Element node to its associated :Sensor nodes, retrieving the unique deviceID for each sensor.

Static parameters required for the physical degradation model would then be retrieved from their respective sources. For this study, we assume these parameters are stored as properties on the :Element nodes within the Neo4j graph. For each structural beam asset (elementID) the parameters include: Geometric Properties: beam.width (b), beam.height (h), beam.spanLength (L). Material Properties: beam.yieldStrength (σ_y), beam.fractureToughness (K_{IC}), beam.paris_C, beam.paris_m. Operational Parameters: beam.initialCrackLength (α_0), beam.cyclesPerDay (n_c). Provenance Data: beam.installationDate, from which "Days in Service" (D) is calculated.

Using the retrieved deviceIDs, queries would be sent to a time-series database (InfluxDB) to fetch dynamic measurements such as Humidity (H), Temperature (T), and Load (P) over a given time window. The combination of this static and dynamic data forms the complete feature vector required for real-time prediction.

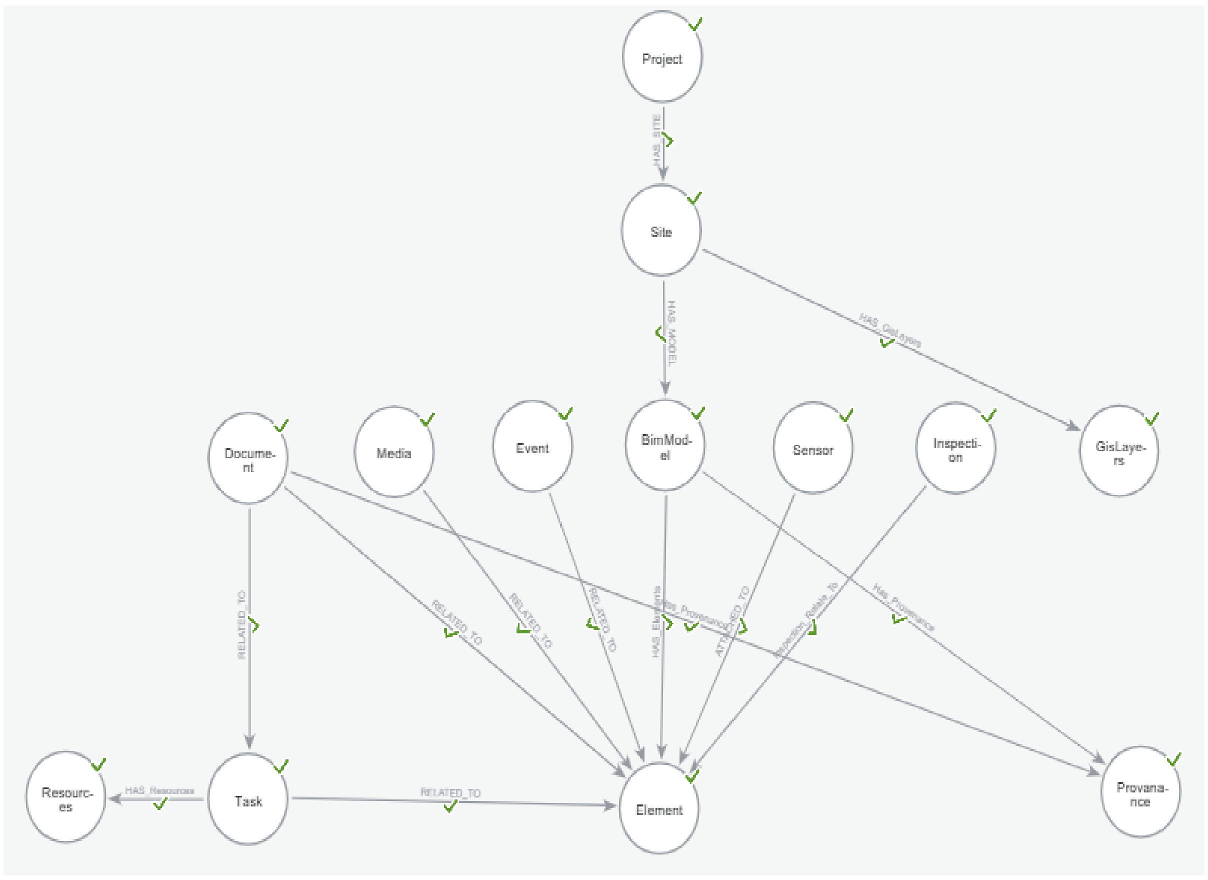


Figure 3: Civil construction project graph model.

5.2. Methodology for Synthetic Data Generation and Model Training

This study validates the proposed predictive models using a synthetically generated dataset. This approach allows for the creation of a balanced and comprehensive dataset to rigorously train and test the machine learning algorithms.

A dataset of 2000 observations was generated. A subset of parameters (Table 1) were sampled from uniform distributions and the remaining parameters were derived according to Equations (1–11). Two distinct ranges were used to create a class-imbalanced scenario reflective of reality:

1. Range 1 (Inspection = 0): These parameters were chosen to simulate beams operating under normal, healthy conditions where an inspection is not required.
2. Range 2 (Inspection = 1): These parameters were chosen to simulate beams under higher stress or with more advanced degradation, representing conditions where an inspection is required according to the binary rule in Equation (12).

To address the class imbalance inherent in generated datasets (where non-failure cases far outnumber failure cases), the SMOTE (Synthetic Minority Over-sampling Technique) was applied to the training data. The default sampling_strategy='auto' was used, which oversamples the minority class until it has the same number of instances as the majority class, resulting in a balanced 1:1 class ratio. The number of nearest neighbors used to generate synthetic samples, k_neighbors, was also kept at its default value of 5. This balancing technique helps prevent the machine learning models from being biased towards the majority class and improves their ability to detect the rare "inspection required" events [15].

Table 1
Data ranges to sample from Uniform distribution to generate dataset

Parameter name	Parameter description	Range 1	Range 2
b	Beam width, m	[0,2 … 0,5]	[0,05 … 0,3]
h	Beam height, m	[0,3 … 0,8]	[0,05 … 0,4]
L	Beam Length, m	[0,5 … 8,0]	[0,5 … 8,0]
yield_strength	stress before plastic deformation, Pa	[250e^{6} … 450e^{6}]	[250e^{6} … 450e^{6}]
K_{IC}	Fracture toughness, Pa/m	[100e^{6} … 300e^{6}]	100e^{6} … 300e^{6}]
α_0	Initial crack length, m	[1e^{-5}… 1e^{-3}]	[5e^{-4} … 5e^{-3}]
P	Load, kH	[100… 50e^{3}]	[3×10^{4} … 10e^{4}]
cycles_per_day	Cycles per day	[100 … 20e^{3}]	[100 … 20e^{3}]

The balanced dataset was then used to train two machine learning classifier algorithms: a Random Forest Classifier (RF) and an Extreme Gradient Boosting Classifier (XGBoost), with the goal of predicting the binary outcome defined in Equation (12).

5.3. Experiment setup

The models were developed and trained using the Python programming language in the following software environment: PyTorch (version 2.7.0 + cpu), NumPy (version 2.2.4), Pandas (version 2.2.3), Scikit-learn (version 1.6.1), Matplotlib (version 3.10.1). Construction graph model was developed and stored in Neo4j desktop (version 2.0.3). Time series data loaded from InfluxDb.

6. Results and discussion

Figure 4 presents the generated dataset characteristics. In Figure 4(a), 4,324 beams are classified as No Inspection Required and 676 as Requires Inspection based on safety thresholds. Figure 4(b) shows the safety factor (SF) distribution, with the dashed line at SF = 2.0 marking the failure threshold (SF < 2.0 is inspection required). Figure 4(c) tracks final crack length (afinal) over time, revealing degradation after ~2000 days.

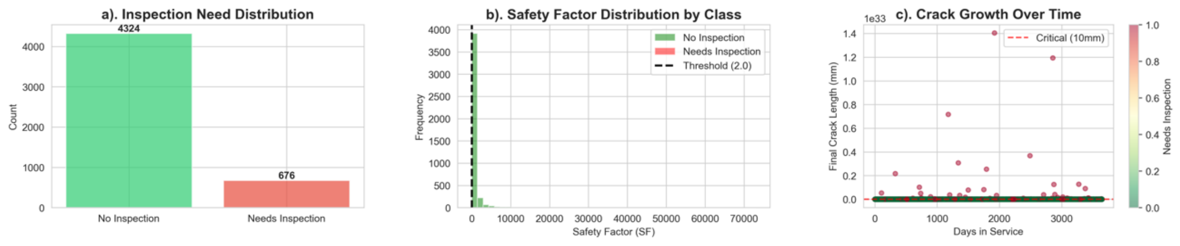


Figure 4: Dataset class, safety factor distributions and crack growth over time.

Figure 5 visualizes physical relationships in the dataset. Red points denote “Need Inspection” and green “No Inspection.” Figure 5(a) shows applied load (P) vs. bending stress (σ): loads < 30 kN are mostly safe; > 60 kN correlate with inspection need. Figure 5(b) shows crack growth rate vs. daily cycles – low cycles (< 5,000/day) show low growth, while > 15,000/day strongly increase da/dN. Figure 5(c) links corrosion to humidity—low humidity (10–40 %) causes minor corrosion; high humidity (70–95 %) accelerates it.

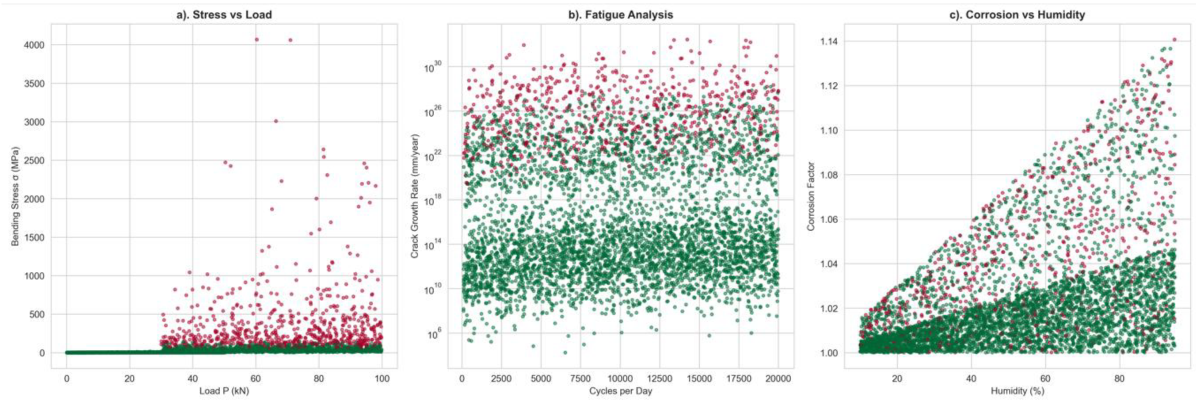


Figure 5: Physical analysis of the relationships in the generated dataset.

Figure 6 confirms the dataset's suitability for ML training. The XGBoost classifier misclassified two "Needs Inspection" cases (recall = 0.9901), while the Random Forest (RF) achieved perfect accuracy (Accuracy = 1.0, F1 = 1.0). Hence, Algorithm 1 was executed using the RF model.

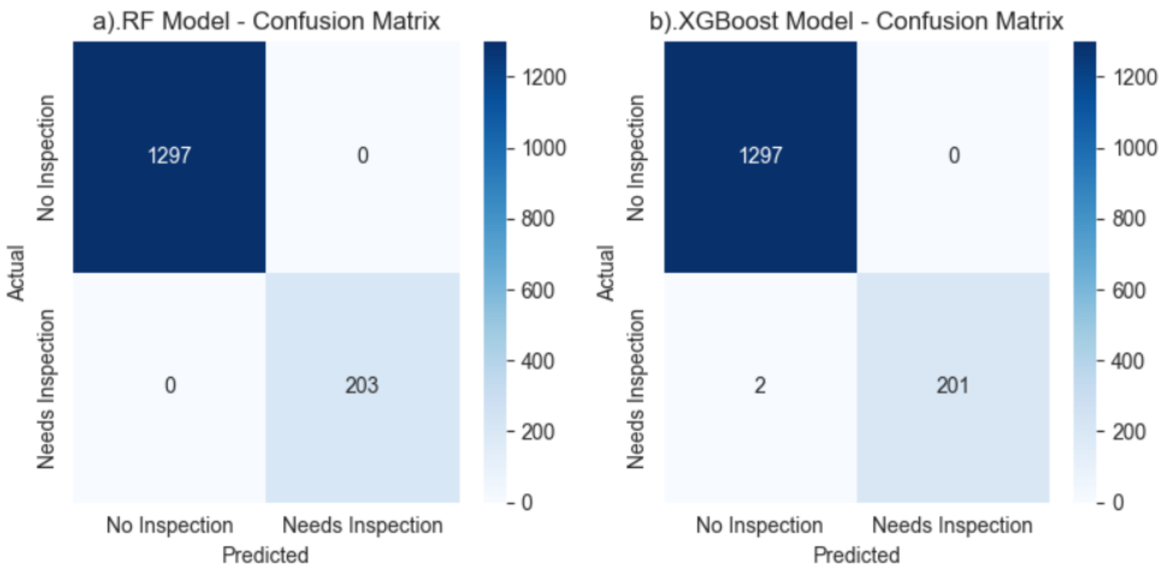


Figure 6: Confusion matrix: (a) RF classifier; (b) XGBoost classifier.

Table 2 records less Accuracy of XGBoost classifier caused by two missed cased as was illustrated on Figure 6.

Table 2
Evaluation results for trained machine learner classifiers

Metric	Random Forest	XGBoost
Accuracy	1	0,9987
F_1	1	1

Therefore, Algorithm 1 is executed with the trained Random Forest model.

The forecasted with Algorithm1 inspections requirements are illustrated on Figure 7. Figure 7 (a) depicts the probability of the beam requiring an inspection over the forecasted time horizon. The red dashed horizontal line indicates the decision threshold (0.5); when the probability of requiring inspection exceeds this threshold, an inspection is recommended. There are three peaks exceeding the threshold, each lasting for a short period before the probability drops: Early in the timeline (near t = 0). At approximately Year 2 and Year 4. On Figure 7 (b) the plot shows the change in the safety factor (SF) over time, with the red dashed line denoting the minimum safety factor threshold (SF = 2.0). When t = 2 years, 4 years SF approaches dips below the critical threshold coinciding with the inspected probability of the beam requiring an inspection peaks in Fig 7. (a) chart. Crack propagation has a spike near the start of the graph (Figure 7 (c)). The crack length grows beyond 10mm, crossing into the critical failure zone early in the forecast. After this initial crack growth, no further crack propagation is observed until Year 4, where another spike is visible. This second spike also crosses into the Critical Zone. Most of the time horizon sees stable or negligible crack growth—suggesting limited degradation during those intervals.

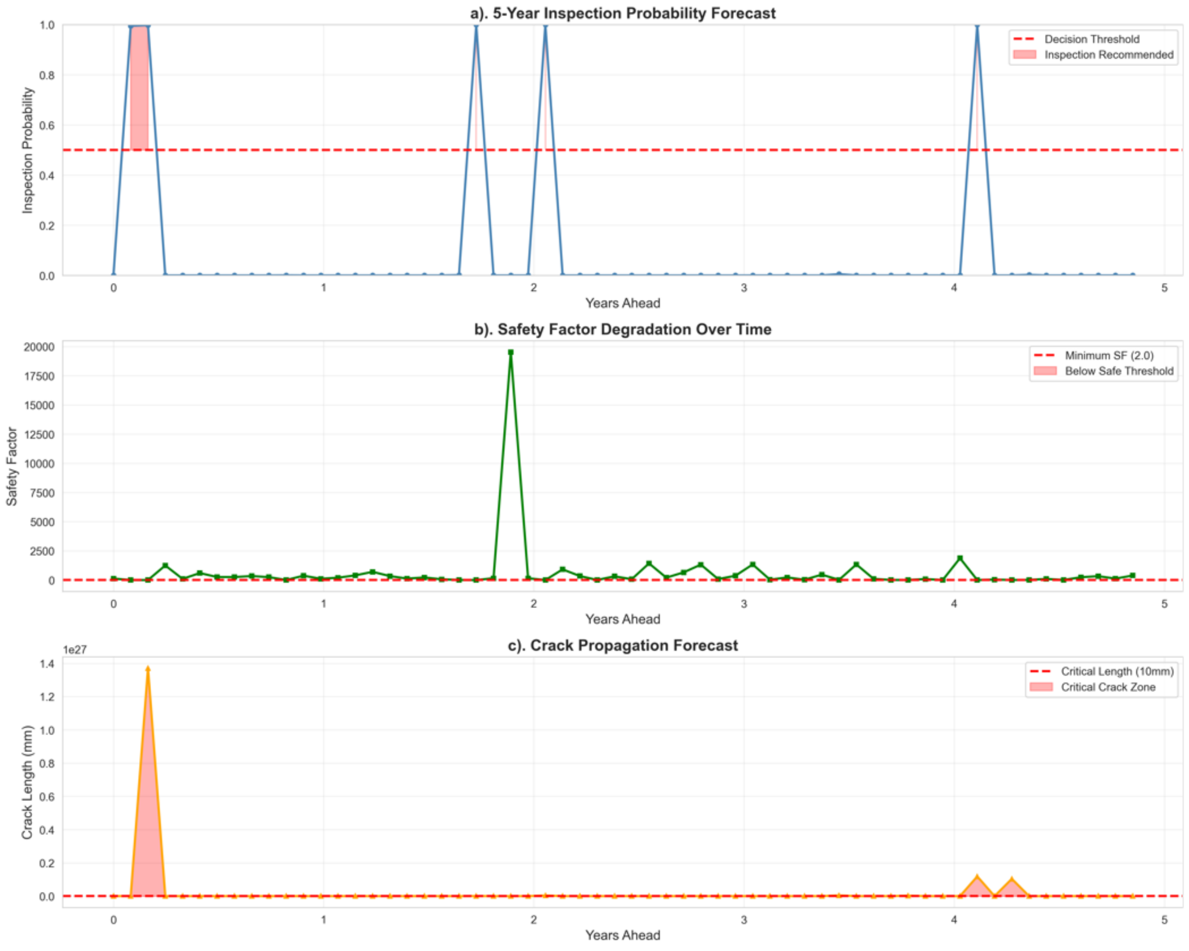


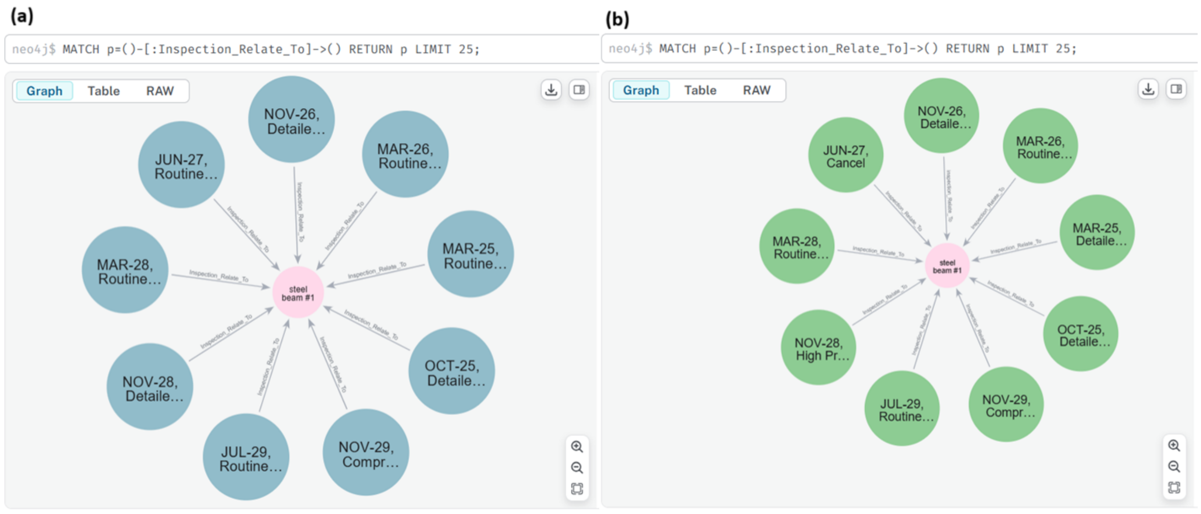
Figure 7: Framework forecasts: (a) 5 years inspections probability forecast; (b) safety factor degradation forecast; (c) crack propagation forecast.

The results of the integration of the predictions for planned inspection are illustrated on Figure 8 (b) and include the following:

1. The static plan's first inspection is a routine visual check on March 2025. However, our predictive model (Figure 7.a) shows a high probability (>0.5) of inspection need starting earlier, driven by a significant initial crack propagation event (Figure 7.c). The framework automatically generates a task for a new, unscheduled "Detailed Inspection" in March 2025, specifically focused on crack assessment.
2. The static plan calls for a 'Detailed Inspection' on November 2026. Our model confirms the need for an inspection around this time, with the probability peaking in late 2026 (approximately Year 2). However, our framework provides additional insight: the driver for this inspection is the Safety Factor (SF) below the critical threshold of 2.0 (Figure 7.b). The scheduled 'Detailed Inspection' for November 2026 is validated and confirmed as necessary.

The framework will update the description, specifying to focus on the factors affecting the safety factor.

3. The static plan includes a 'Routine Visual' inspection on June 2027. Our predictive model shows a very low probability of inspection need throughout 2027. The safety factor is above the threshold, and crack growth during this period. The framework updates the inspection's description recommending that the inspection scheduled for June 2027 is unnecessary and can be safely deferred.
4. The static plan schedules a 'Detailed Inspection' for November 2028. Our model concurs, showing a sharp spike in inspection probability around Year 4 (~ October 2028). The underlying drivers are identified as another significant crack propagation event and a corresponding dip in the safety factor. The framework updates the inspection's description by setting a high-priority.


(a)

```
neo4j$ MATCH p=()-[:Inspection_Relate_To]->() RETURN p LIMIT 25;
```

(b)

```
neo4j$ MATCH p=()-[:Inspection_Relate_To]->() RETURN p LIMIT 25;
```

Figure 8: Inspections plan: (a) calendar-based; (b) updated with forecasted inspection needs.

7. Conclusions

This research developed and validated a novel framework for predictive inspection scheduling in construction projects. The core contribution is a hybrid physics-machine learning (ML) methodology, where a physics-based degradation model of a structural beam generated a high-fidelity synthetic dataset for training ML classifiers.

Applying the forecasting algorithm to a real-world, static inspection plan demonstrated the framework's practical value. It dynamically identified a new high-risk period requiring an added inspection, validated necessary scheduled inspections with causal insights, and flagged a low-risk period where an inspection could be safely deferred-optimizing resources while enhancing safety and efficiency.

The main limitation is that the ML models were trained on synthetic data derived from physical models, which cannot fully represent real-world noise, sensor errors, or operational variability. The study was limited to one asset type (a structural beam) and specific degradation modes (corrosion and fatigue), excluding others such as fire or impact damage.

Future work will expand the framework's scope by developing a library of degradation models for various asset types and incorporating explainable AI (XAI) methods, such as SHAP, to increase transparency and user trust in predictive decisions.

Declaration on Generative AI

The authors has not employed any Generative AI tools.

References

[1] M. Mahmoodian, F. Shahrivar, S. Setunge, S. Mazaheri, Development of digital twin for intelligent maintenance of civil infrastructure, Sustainability 14 (14) (2022). doi:10.3390/su14148664.

[2] H. H. Hosamo, H. K. Nielsen, A. N. Alnmr, P. R. Svennevig, K. Svidt, A review of the digital twin technology for fault detection in buildings, Frontiers in Built Environment 8 (2022). doi:10.3389/fbuil.2022.1013196.

[3] Y. B. S. Reddy, M. Z. Kangda, E. N. Farsangi, Integration of building information modeling and machine learning for predictive maintenance, in: E. N. Farsangi, M. Noori, T.y Yang, V. Sarhosis, S. Mirjalili, M. J. Skibniewski (Eds.), Digital Transformation in the Construction Industry, Woodhead Publishing, Sawston, Cambridge, 2025, pp. 361–378. doi:10.1016/B978-0-443-29861-5.00017-2.

[4] I. Abdullahi, S. Longo, M. Samie, Towards a distributed digital twin framework for predictive maintenance in industrial internet of things (IIoT), Sensors 24 (8) (2024). doi:10.3390/s24082663.

[5] A. E. Bondoc, M. Tayefeh, A. Barari, Learning phase in a LIVE digital twin for predictive maintenance, Autonomous Intelligent Systems 2 (1) (2022). doi:10.1007/s43684-022-00028-0.

[6] S. W. Yang, Y. Lee, S. A. Kim, Design and validation of a real-time maintenance monitoring system using BIM and digital twin integration, Buildings 15 (8) (2025). doi:10.3390/buildings15081312.

[7] M. S. Khan, Data-driven digital twin-based smart tunnel maintenance system, Intelligent Geoengineering (2025). doi:10.1016/j.ige.2025.09.002.

[8] C. Hadjichristofi, S. Diochnos, K. Andresakis, V. Vescoukis, Using time-series databases for energy data infrastructures, Energies 17 (21) (2024). doi:10.3390/en17215478.

[9] J. M. Gere, B. J. Goodno, Mechanics of Materials, 5th ed., Brooks/Cole, Pacific Grove, CA, USA, 2001.

[10] ASM Handbook, volume 1: Properties and Selection: Irons, Steels, and High Performance Alloys, ASM International, Novelty, Ohio, USA, 1990.

[11] T. L. Anderson, Fracture Mechanics: Fundamentals and Applications, Chapman & Hall/CRC, Boca Raton, USA, 2005.

[12] L. F. F. Ricardo, T. H. Topper, L. C. H. Ricardo, C. A. J. Miranda, Crack propagation in the threshold stress intensity region: a short review, in: Proceedings of the XIX International Colloquium on Mechanical Fatigue of Metals, Springer, Cham, 2019, pp. 175–180. doi:10.1007/978-3-030-13980-3_23.

[13] W. D. Callister Jr., D. G. Rethwisch, Materials Science and Engineering: An Introduction, John Wiley & Sons, Hoboken, New Jersey, 2020.

[14] ASM Handbook, volume 19: Fatigue and Fracture, ASM International, Novelty, Ohio, USA, 1996.

[15] D. Gobov, O. Solovei, Approaches to improving the accuracy of machine learning models in requirements elicitation techniques selection, in: Proceedings of the International Conference on Computer Science, Engineering and Education Applications, Springer Nature, Cham, 2023, pp. 574–584. doi:10.1007/978-3-031-36118-0_51.