

AI-Driven Track Asset Lifecycle Prediction for Indian Railways Using Laser QR, Weather and Service Loads

Malar K¹, Nimitha Shetty², Shashwath K S² & Yashashwini R²

¹Assistant Professor, Department of Information Science and Engineering
Sri Sairam College of Engineering, Bengaluru

²Final Year, BE (Information Science and Engineering), Sri Sairam College of Engineering, Bengaluru

DOI: doi.org/10.34293/iejcsa.v4i5.175

Abstract - Railway track assets such as rail sections, fastenings, sleepers and welds are still inspected largely on fixed calendar schedules, so degradation that depends on traffic and climate is detected late or not at all. This paper presents Smart Track AI, a web-based platform that links each track asset to a laser-marked QR tag, collects service-load and weather telemetry at the point of inspection, and returns an estimate of Remaining Useful Life (RUL) together with a three-level health status. The platform couples a browser-side camera scanner with a Flask inference service that serves a pre-trained Random Forest regressor. Because real Indian Railways failure logs were not available to the authors, the prediction pipeline was evaluated on 3000 synthetic asset records whose degradation follows load-, corrosion- and thermal-fatigue-inspired relationships. On the held-out set the Random Forest reached an MAE of 5.20 months and an RZ of 0.945, and mapping the predicted RUL to Optimal, Warning and Critical bands gave 92.3% status accuracy; gradient boosting was marginally more accurate. The results demonstrate that the end-to-end workflow is feasible; validation on real railway data remains future work.

Keywords: predictive maintenance, remaining useful life, railway track assets, QR code, Random Forest, Flask, Indian Railways

INTRODUCTION

Indian Railways operates one of the densest rail networks in the world, and the safe running of its trains depends on thousands of kilometres of track whose condition changes continuously. Rails wear under repeated wheel contact, fastenings loosen under vibration, sleepers and ballast deteriorate in wet conditions, and welds accumulate fatigue as temperature swings through the year. Most of this work is currently managed through periodic walking inspections and time-based replacement rules. Time-based rules are simple to administer, but they treat a lightly used loop line and a heavily loaded freight corridor as if they aged at the same pace, which either wastes serviceable material or leaves worn components in service for too long.

Condition-based and predictive maintenance aim to replace the calendar with evidence. The idea is to estimate, from the operating history of an asset and the environment around it, how many months of safe service remain, and to schedule intervention accordingly. Two practical obstacles stand in the way on a large network: first, each asset must be reliably identified in the field so that readings are attached to the right

record; second, the predictive model must be reachable by the person standing next to the asset, not only by an analyst in an office.

Maintenance of this kind is also constrained by access. Track possession time is limited because running lines carry heavy traffic, so inspection and repair work must be planned into short windows. A reliable estimate of which assets are closest to the end of their useful life helps planners spend those windows where they matter. At the same time, any decision tool used in a safety-critical setting must remain an aid to the responsible engineer rather than a replacement for judgement, and the platform described here is designed with that role in mind: it ranks and flags, and a person decides.

This work addresses both obstacles with a single lightweight platform. Our contributions are as follows. First, we present a field workflow in which each track asset carries a laser-marked QR tag that is read by an ordinary smartphone or laptop camera, removing manual entry of asset identifiers. Second, we describe a modular web architecture that joins optical scanning, a Flask REST service, a stored Random Forest model, a dashboard and an audit log. Third, we use a feature set that combines service loads (axle load, cumulative traffic, speed, curvature) with weather exposure (rainfall, humidity, temperature swing) to predict RUL in months and a three-level health status. Fourth, we provide a reproducible synthetic-data evaluation comparing four regressors, with transparent reporting of where the proposed model is and is not the best choice.

The remainder of the paper is organised as follows. The literature review surveys remaining useful life prognostics, ensemble regression and optical identification of assets, and the research gap and problem statement follow from it. The proposed system, its architecture, and the methodology and workflow are then described, after which the implementation and evaluation are presented together with a discussion of limitations. The paper ends with the conclusion and future work.

LITERATURE REVIEW

A. Remaining Useful Life Prognostics

RUL prediction estimates the time remaining before an asset can no longer perform its function. Surveys of machinery prognostics organise the field into physics-based models, data-driven models and hybrids, and note that data-driven approaches have grown fastest because sensor and inspection data have become cheaper to collect [4]. Public run-to-failure benchmarks, notably the turbofan degradation simulations released by NASA, have given researchers a common way to compare data-driven RUL methods when real failure histories are scarce [3]. We follow the same spirit here: where field failure data cannot yet be shared, a documented simulator allows the pipeline itself to be tested.

B. Ensemble Regression for Tabular Telemetry

Tabular operating data of mixed scale and with nonlinear interactions is a natural fit for tree ensembles. Random Forests average many decorrelated decision trees trained on

bootstrap samples, which reduces variance and gives a built-in measure of feature importance [1]. Gradient boosting builds trees sequentially, each correcting the residual error of the ensemble so far, and often attains slightly lower error at the price of more careful tuning [5]. Both are available in scikit-learn [2], which keeps the deployment footprint small. General treatments of these methods are given in [6].

C. Optical Identification of Assets

QR codes encode data in a two-dimensional matrix with built-in error correction, so that a symbol remains readable even when part of it is damaged or dirty [7]. This property matters on track, where dust, grease and paint are unavoidable. Direct part marking by laser engraving produces a code that is part of the metal itself rather than a label that can peel away. In this paper the QR code carries only an asset identifier; all condition data live in the database, so a damaged tag can be re-issued without losing history.

D. Track Degradation and Maintenance Modelling

Work on rail infrastructure has concentrated mainly on track geometry rather than on individual components. Soleimanmeigouni et al. review how geometry degradation is analysed for maintenance planning [8]. Andrade and Teixeira stress that geometry degradation is inherently uncertain, even on high-speed lines, and that maintenance decisions should account for that uncertainty [9]. Lasisi and Attoh-Okine show how principal component analysis can condense many geometry parameters into a single track quality index [10].

These studies draw on measurements collected by inspection vehicles, so they work with rich and regularly sampled signals. They say little about discrete components such as fastenings, sleepers and welds, which are in practice inspected on foot, and they do not address how a field engineer would identify the asset and obtain a prediction at the point of inspection. That is the setting targeted in this paper.

E. Statistical and Deep-Learning Approaches to RUL

Statistical data-driven methods, including stochastic-process and filtering models of degradation, are surveyed in [11]. With the spread of benchmark data, neural networks have become common as well; for example, long short-term memory networks have been applied to RUL estimation on the NASA turbofan data [12].

Sequence models of this kind need long, continuous histories and a sizeable number of run-to-failure examples. Neither is available for passive track components that are visited at irregular intervals, which is why compact tree ensembles on a snapshot of operating conditions are a sensible first step here, with sequence models left for later work.

F. Predictive Maintenance in Railways and Optical Identification

Carvalho et al. review machine-learning methods for predictive maintenance across industries and find tree ensembles, support vector machines and neural networks among

the most frequently applied [13]. In the railway domain, Ghofrani et al. survey big-data applications, including maintenance and asset management [14]. Instrumenting every component over thousands of kilometres remains costly, which motivates the choice here to identify assets cheaply and to borrow traffic and weather context from existing records.

On the identification side, laser direct part marking produces 2D codes that are part of the component itself, and their recognition under severe abrasive conditions has been studied experimentally [15]. This supports the use of laser-marked QR tags on track, although field durability on rail steel remains one of the open items listed under future work.

G. Summary of Related Work and Positioning

In summary, RUL research is mature for rotating machinery and engines [3], [4], [11], [12], track research has focused on geometry measured by inspection vehicles [8]-[10], and railway maintenance surveys describe the potential of data-driven methods [13], [14] without prescribing a low-cost field workflow. The present work does not propose a new learning algorithm. Its contribution is to combine laser-marked QR identification, service-load and weather context, a compact ensemble model and an audit log into one workflow that a field engineer can use on a phone, together with an honest synthetic-data baseline.

RESEARCH GAP

Most published RUL work targets rotating machinery or aircraft engines, where continuous multichannel sensors are available. Track assets differ in three ways. They are distributed over very long linear distances, so instrumenting every asset is uneconomical. They are passive components with no internal sensors of their own. And their degradation is driven by external factors, namely traffic and climate, which can be obtained from operational and meteorological records rather than from the asset itself.

These differences motivate a design that identifies the asset cheaply, borrows context from existing records, and reserves expensive measurement for inspection visits. The gap addressed here is therefore not a missing learning algorithm but the lack of a usable workflow that brings identification, data capture, prediction and logging together. A second gap is the scarcity of field failure data that can be shared, which means that any new pipeline needs a documented simulator and an honest baseline for its accuracy under controlled conditions before it can be tested on real track. The contribution of this paper is accordingly the integration of identification, data capture, prediction and logging into one usable workflow, together with such a baseline.

PROBLEM STATEMENT

Let x denote the feature vector of Table I for one asset at the time of inspection and let y be its remaining useful life in months. The goal is to learn a function f such that $f(x)$ approximates y with small absolute error, and then to map $f(x)$ to one of three maintenance

categories. Two practical requirements shape the design. The function must be cheap enough to evaluate on demand during a scan, and it must be explainable enough that an engineer can see which factors drive a low estimate. These requirements favour compact ensemble models over deep networks for a first deployment on tabular data of this size.

The setting differs from the more common sensor-stream prognostics problem in one respect: the model is queried at irregular, human-triggered moments rather than being fed a continuous signal. A snapshot formulation is therefore natural, and it keeps the scanner payload small. The price is that temporal patterns, such as an accelerating rate of wear, are not seen by the model unless they are summarised in the features; the conclusion returns to this point.

PROPOSED SYSTEM

Guided by the gaps above, the proposed system, Smart Track AI, links each track asset to a laser-marked QR tag that is read by an ordinary smartphone or laptop camera, collects service-load and weather telemetry at the point of inspection, and returns an RUL estimate together with a three-level health status of Optimal, Warning or Critical. The estimate comes from a Random Forest regressor that is served by a Flask application, and the platform is intended as decision support in which the engineer, not the model, makes the final call.

The platform is organised into five modules. Security and access control is provided by a login page and a shared base template that give role-based authentication; the base template also carries the navigation and session-dependent layout so that every view enforces the same access rules. Optical data acquisition is handled by a scanner page and its script, which access the device camera, decode QR codes and barcodes in real time, and post the asset identifier and the accompanying readings to the backend, so that no identifier is typed by hand. The predictive analytics pipeline is the Flask application, which receives a JSON payload of operating conditions, passes the features to the stored Random Forest and returns an RUL estimate with a status label. Visualisation and activity logging are supported by a dashboard that shows current telemetry, notifications and health scores, and by an activity page that keeps a time-stamped audit trail of scans, inspections and prediction outcomes. Finally, asset management and tag generation allows administrators to register new assets and generate the corresponding QR tags that are later laser-marked and fixed to the asset.

Two roles use the platform. An administrator logs in, registers a new asset with its identifier and installation date, and generates the tag image for laser marking. A field engineer logs in on a phone or laptop, opens the scanner page, points the camera at a tag and confirms the readings gathered for that visit. Within a short time the dashboard shows the status band and the RUL estimate, and the entry appears at the top of the activity log. A supervisor reviewing the log can filter by asset or date to see how an asset's estimated life

has changed from one inspection to the next, which is the first step toward trend-based maintenance.

SYSTEM ARCHITECTURE

Smart Track AI is a full-stack application split into a browser-side presentation tier and a server-side processing tier, following a model–view–controller organisation. The presentation tier consists of HTML5 templates, a custom style sheet and a JavaScript module that drives the camera. The processing tier is a Python Flask application that exposes REST routes, maintains user sessions, talks to the data store and runs model inference. Fig. 1 shows the layers and the direction of data flow.

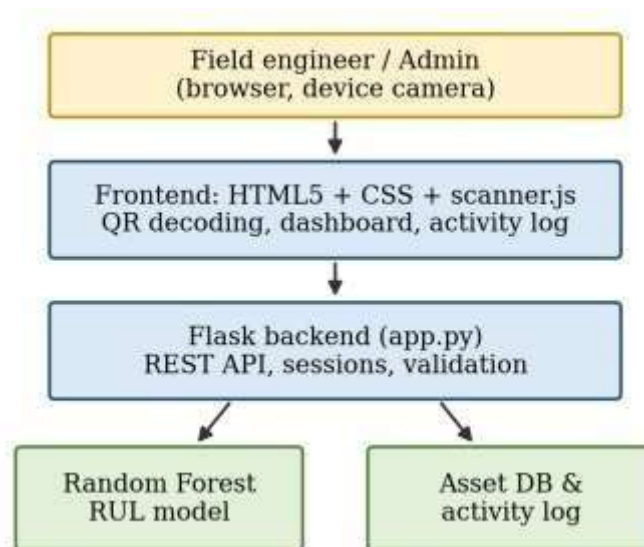


Figure 1 Layered architecture of the Smart Track AI platform.

The data store is organised around four kinds of record. A user record holds credentials and a role. An asset record holds the identifier, location description and installation date. A reading record holds one set of telemetry values captured at a scan, linked to its asset and user. A prediction record holds the RUL, the status band and the time stamp, linked to the reading that produced it. Because each prediction points back to the exact reading and to the signed-in user, an auditor can later reconstruct why an asset was flagged.

For deployment the application is packaged with a requirements file that lists Flask, a production WSGI server (Gunicorn), NumPy, scikit-learn and a time-zone library used to stamp records consistently. Because the service is stateless apart from the database and the loaded model, several worker processes can run behind a reverse proxy when load grows. Administrators generate QR tags from the registration page; each tag encodes only the asset identifier, in line with the design principle described in the literature review.

METHODOLOGY AND WORKFLOW

A. End-to-End Workflow

Fig. 2 traces one asset through its life in the system. An administrator registers the asset and produces its tag. During an inspection the engineer scans the tag, the scanner module attaches the current readings and sends them to the server, and the server validates the request, loads the asset record and runs the model. The predicted RUL is converted to a status band and returned to the dashboard while a record is written to the activity log. Because identification, data capture and prediction occur in one interaction, the delay between inspection and decision is limited to a single network round trip plus model inference.

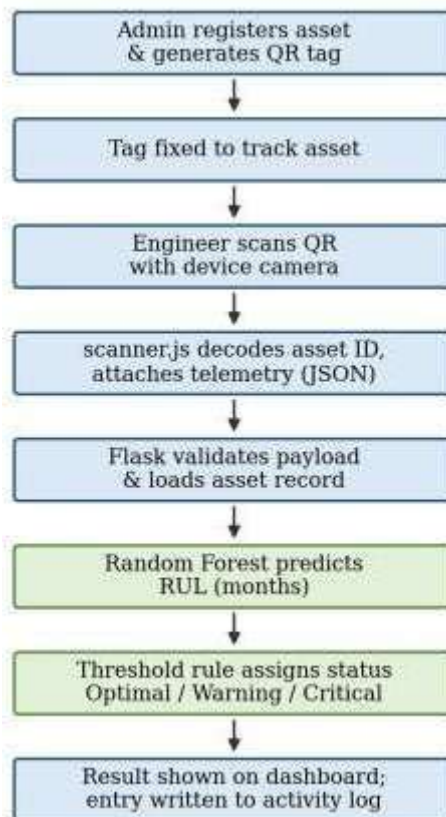


Figure 2 Flow from asset registration to a logged prediction.

B. Input Variables

The model uses two groups of inputs, summarised in Table I. Service-load variables describe how hard the asset has been worked, and weather variables describe how aggressive its environment has been. Asset age is included because ageing mechanisms such as corrosion progress with time even on lightly used track.

Table 1 Input Features and Target

Feature	Unit	Meaning
asset_age	months	Time since installation
cumulative_MGT	MGT	Traffic carried to date
axle_load	tonne	Mean axle load
avg_speed	km/h	Mean running speed
rainfall	mm/yr	Annual rainfall exposure
humidity	%	Mean relative humidity
temp_swing	°C	Daily temperature range
curve_flag	0/1	Asset on a curved section
RUL (target)	months	Remaining useful life

C. Synthetic Dataset

No operational failure logs from Indian Railways were available to us, so the learning pipeline was exercised on a synthetic data set of 3000 asset records that we generated with a fixed random seed. We state this plainly because the numerical results in the evaluation describe how well the pipeline recovers a known degradation law; they do not describe performance on real track. Each feature was drawn from a plausible range (for example, asset age between 1 and 180 months and mean axle load around 20.5 t). Degradation was built from three contributions. A wear term grows with cumulative traffic, with axle load raised to the third power, with the square root of speed, and is amplified on curves. A corrosion term couples age with rainfall and the square of humidity. A thermal-fatigue term couples age with temperature swing. With W , C and T denoting the normalised wear, corrosion and thermal terms, the damage index and RUL are

$$D = 1.1 W + 0.45 C + 0.25 T \quad (1)$$

$$RUL = 140 \cdot \exp(-1.15 D) + \varepsilon \quad (2)$$

where ε is zero-mean Gaussian noise whose spread increases with RUL. The resulting RUL ranges from

0.7 to 140.2 months with a mean of 47.3 months. Using the band definitions given under health status mapping below, the data contain 2292 Optimal, 654 Warning and only 54 Critical records, so the Critical class is rare. The functional form and constants are modelling assumptions of ours, not measured railway relationships, and they should be replaced by site data when available.

D. Learning Pipeline

Fig. 3 summarises the pipeline. The data were split 80/20 into a training set of 2400 and a test set of 600 records. Four regressors were trained on identical splits: ordinary linear regression as a baseline, a depth-limited decision tree, gradient boosting and a Random Forest. The Random Forest used 200 trees with default scikit-learn settings, the boosting

model used 200 stages, and the tree was limited to depth ten (Table II). In addition to the hold-out score, five-fold cross-validation over the full data was used to check stability.

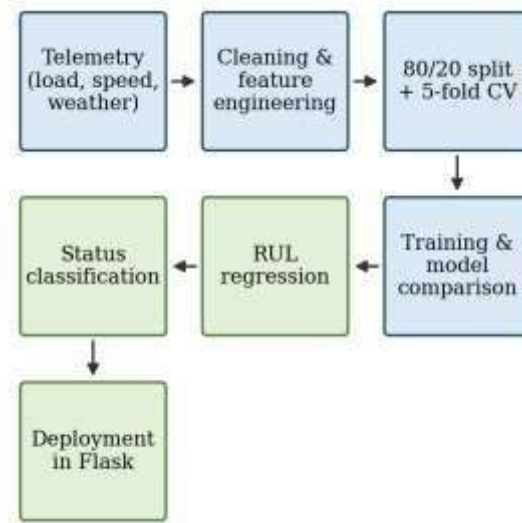


Figure 3 Training and deployment pipeline.

Table 2 Model Settings Used in the Comparison

Model	Setting	Value
Linear Regression	—	Ordinary least squares
Decision Tree	max depth	10
Gradient Boosting	stages	200
Random Forest	trees	200
All models	seed / split	42 / 80:20

Tree-based models are insensitive to monotonic rescaling of inputs, so no feature scaling was applied; the linear baseline was fitted on the raw values for the same reason of simplicity. Preprocessing was limited to range checks and a fixed column order. The test set was held out before any model was fitted and was not used to choose settings, which avoids leakage of test information into training.

Accuracy was measured with mean absolute error, root mean squared error and the coefficient of determination:

E. Health Status Mapping

$$MAE = (1/n) \sum |y_i - \hat{y}_i| \quad (3)$$

$$RMSE = \sqrt{(1/n) \sum (y_i - \hat{y}_i)^2} \quad (4)$$

$$R^2 = 1 - \sum (y_i - \hat{y}_i)^2 / \sum (y_i - \bar{y})^2 \quad (5)$$

Maintenance staff need an action category rather than a raw number, so the predicted RUL is mapped to a status band: Critical below 6 months, Warning from 6 up to 24 months, and Optimal at 24 months or more. These thresholds are illustrative defaults used

in the experiment; an operating railway would set them from inspection intervals and the lead time needed to procure and fit replacements. Fig. 4 shows the decision rule applied to every prediction before the result is displayed.

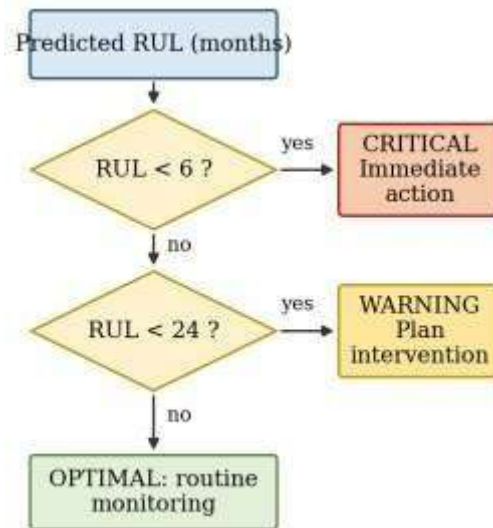


Figure 4 Decision logic that converts predicted RUL into a status band and a recommended action.

F. Evaluation Protocol and Ablation Design

Beyond the headline comparison, three further experiments were run on the same split. An ablation removed one feature group at a time (weather, service load, or asset age) and retrained the Random Forest, to measure how much each group contributes. A sensitivity study varied the number of trees from 10 to 400. Finally, runtime cost was measured: training time, serialised model size and single-record prediction latency, because the platform answers one scan at a time and response delay is part of the user experience. All experiments use the same random seed, so the results reported in the next section can be regenerated exactly.

IMPLEMENTATION AND EVALUATION

A. Implementation

The backend is a single Flask application. At start-up it loads the serialised Random Forest once, so each request pays only the cost of inference. A prediction route accepts a JSON body containing the asset identifier and the feature values of Table I, checks that required fields are present and numeric, builds the feature vector in the same column order used during training, and returns the RUL estimate and status as JSON (Table III). Keeping the column order identical between training and serving is essential, because a silent mismatch would produce plausible but wrong numbers.

Table 3 Request and Response Fields of the Prediction Service

Field	Dir.	Content
asset_id	in	Identifier decoded from the QR tag
load / traffic fields	in	Axle load, MGT, speed, curve flag
weather fields	in	Rainfall, humidity, temp. swing
asset_age	in	Months in service
rul_months	out	Predicted remaining life
status	out	Optimal, Warning or Critical
timestamp	out	Server time of the prediction

On the client, the scanner page requests camera permission, decodes frames continuously and, on a successful read, submits the payload without further input. The dashboard polls the server and shows health scores and notifications, while the activity page lists earlier scans with their time stamps and model outputs, giving supervisors an audit trail. Pages inherit a common base template so that navigation, session state and security headers are consistent.

Data handling deserves care. Scan payloads travel from a mobile browser to the server and therefore should be sent over an encrypted connection only. Inference requests are validated before use so that malformed or out-of-range values are rejected instead of being passed to the model, and every accepted request is logged with the signed-in user. Role separation keeps registration and tag generation limited to administrators, while field engineers can scan and view results but cannot alter asset records.

B. Illustrative Scenarios

To show how outputs read in practice, three hypothetical assets were passed to the trained model (Table IV). They are invented for illustration and do not correspond to real track. Asset A is old, on a curve, carries heavy axle loads and sits in a wet climate; the model returns 3.3 months and the Critical band. Asset B is newly laid and lightly loaded in a dry climate, and receives 96.7 months and Optimal. Asset C lies between the two, with 20.4 months and a Warning, close to the boundary with Optimal. Such a case is where an engineer's judgement should weigh in, since a small change in input or error in the estimate could move the asset across a band.

Table 4 Three Hypothetical Assets (Model Outputs)

Asset	Age / MGT	Axle (t)	RUL (mo.)	Status
A	150 / 350	23.5	3.3	Critical
B	40 / 60	19.5	96.7	Optimal
C	110 / 220	21.5	20.4	Warning

Remaining inputs: speed 100/70/85 km/h; rainfall 2200/700/1200 mm; humidity 85/55/70%; temp. swing 20/10/14 °C; curve 1/0/1.

C. Regression Accuracy

Table V reports hold-out and cross-validated results. Gradient boosting achieved the lowest error (MAE 4.14 months, RZ 0.964), followed by the Random Forest (MAE 5.20 months, RZ 0.945). Both clearly outperformed the linear baseline (MAE 7.61 months) and the single decision tree (MAE 8.20 months). The linear model's weaker result is expected because the synthetic degradation law is strongly nonlinear in load, humidity and age. Cross-validated RZ values agree closely with the hold-out scores and show small spread, which indicates that the ranking is not an accident of one particular split.

Table 5 Model Comparison on the Synthetic Test Set

Model	MAE	RMSE	R ²	CV R ² (5-fold)
Linear Regression	7.61	9.94	0.882	0.872 ± 0.013
Decision Tree	8.20	10.89	0.859	0.843 ± 0.015
Gradient Boosting	4.14	5.51	0.964	0.963 ± 0.002
Random Forest	5.20	6.80	0.945	0.935 ± 0.007

MAE and RMSE in months.

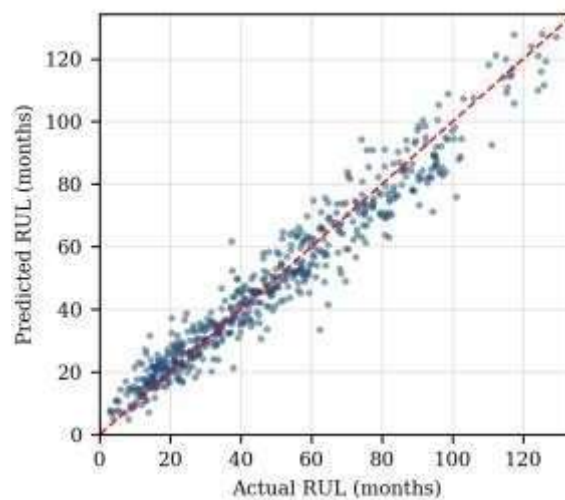


Figure 5 Predicted versus actual RUL for the Random Forest on the test set. The dashed line marks perfect prediction.

Fig. 5 shows that Random Forest predictions follow the diagonal closely across the whole range (residual spread in Fig. 7). The scatter widens slightly for long-life assets, where the simulated noise is largest. We retained the Random Forest as the deployed model despite the small accuracy advantage of boosting, for three reasons: its training is less sensitive to hyper-parameters, it needs no tuning to reach this level of accuracy, and its feature importances are straightforward to explain to maintenance staff. Whether this

trade-off is right for real data is an empirical question that should be re-examined once field records exist.

D. Feature Importance

Fig. 6 ranks the inputs by their mean impurity reduction in the Random Forest. Cumulative traffic carries the largest share (0.58), followed by asset age (0.17), axle load (0.08) and the curve indicator (0.06). Weather variables contribute less individually, with rainfall at 0.05, but they act through interactions with age, which a single importance score understates. These rankings mirror the structure we built into the simulator, so they confirm that the model recovered the planted relationships; they should not be read as findings about real track.

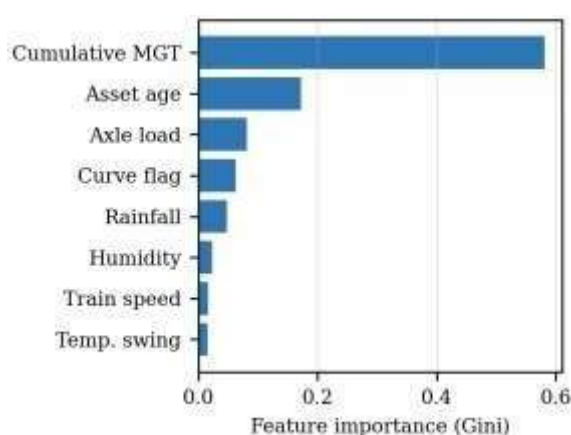


Figure 6 Feature importance of the Random Forest model.

E. Status Classification

Converting predicted RUL into the three bands gave an overall accuracy of 92.3% on the 600 test records. Table VI gives the confusion matrix. Most errors fall between neighbouring bands, which is the benign direction for a maintenance planner: of the 149 Warning assets, 28 were labelled Optimal and one Critical. Only 10 Critical assets appeared in the test set, of which 1 was identified and 9 were reported as Warning, so no reliable conclusion about Critical-class recall can be drawn from this experiment.

Table 6 Confusion Matrix of Status Bands (Rows: Actual)

Actual \ Predicted	Optimal	Warning	Critical
Optimal	433	8	0
Warning	28	120	1
Critical	0	9	1

F. Error Distribution

Fig. 7 shows the distribution of residuals. They are centred close to zero with a standard deviation of 6.8 months. 37.8% of test predictions fell within 3 months of the true

RUL, 58.8% within 5 months and 86.8% within 10 months; the 95th percentile of absolute error was 13.8 months and the largest single error was 28.8 months. These figures are a useful complement to the MAE because they show the tail: a planner who relies on an estimate should know that roughly one prediction in eight can be off by more than ten months even in this controlled setting, and that real data, with unmodelled effects, is likely to be noisier.

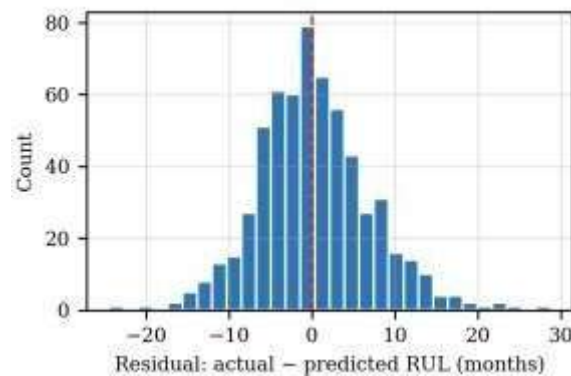


Figure 7 Distribution of test residuals for the Random Forest.

G. Per-Band Error

Error is not uniform across the RUL range. Table VII splits the test MAE by the true status band. Assets in the Warning band, which matter most for scheduling work, were predicted with the smallest error (3.97 months), while long-life Optimal assets showed the largest (5.63 months). This pattern is a direct result of the simulator, where noise grows with RUL, and it is convenient: absolute errors of a few months matter little when thirty or more months remain, but they matter a great deal near the decision threshold, where the model is most accurate. The mean signed residual was 0.30 months, so predictions show no meaningful systematic bias. The ten Critical test cases are too few to support strong claims.

Table 7 Test MAE by True Status Band

True band (months)	n	MAE (months)
Optimal (≥ 24)	441	5.63
Warning (6-24)	149	3.97
Critical (< 6)	10	4.48

H. Ablation of Feature Groups

Table VIII and Fig. 8 show the effect of removing each feature group. Removing the service-load features collapsed the model (MAE 21.15 months, RZ 0.202), which is unsurprising since traffic and axle load drive the wear term in the simulator. Removing asset age raised the MAE to 10.10 months. Removing the three weather variables increased the MAE from 5.20 to 6.68 months, a rise of about 29%, and lowered RZ to 0.898. Weather is therefore a smaller but real contributor, consistent with the premise of this work that

climate exposure should be modelled alongside traffic. Again, the size of each effect depends on the weights we gave each term in the simulator and would have to be re-estimated from real data.

Table 8 Random Forest Ablation Results

Feature set	MAE	R ²
All features	5.20	0.945
No weather	6.68	0.898
No service load	21.15	0.202
No asset age	10.10	0.782

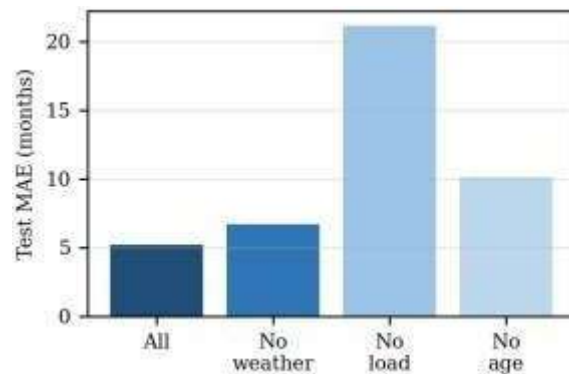


Figure 8 Test MAE of the Random Forest when a feature group is removed.

I. Model Size and Responsiveness

Fig. 9 shows that test error falls quickly up to about 50 trees and then flattens, with 400 trees improving MAE by under 0.05 month over 200. The 200-tree setting is therefore a reasonable compromise. On the machine used for the experiments, fitting the 200-tree model took 1.7 s. Predicting a single record with one worker thread took a median of 10.4 ms (95th percentile 14.5 ms). These timings come from our development environment, not from target field hardware, so they indicate order of magnitude only. The serialised model occupies about 44 MB, which is acceptable for a server but would argue against shipping the model to phones; keeping inference on the server, as the architecture does, avoids that problem. If memory becomes a concern, limiting tree depth or leaf size would reduce the size with a modest accuracy cost.

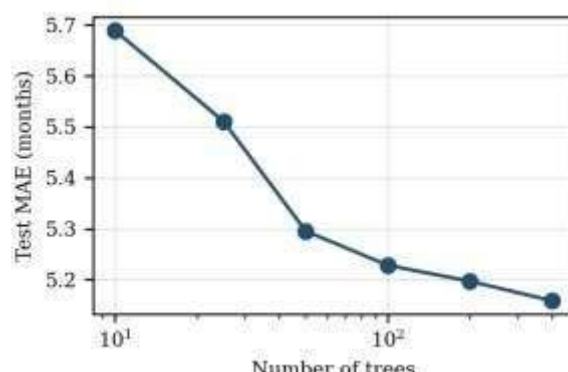


Figure 9 Effect of the number of trees on test MAE.

J. Summary of Findings

Four observations stand out. First, tree ensembles clearly outperformed the linear baseline and the single tree on the nonlinear synthetic task, with gradient boosting marginally ahead of the Random Forest. Second, service-load and age variables dominated predictive power, while weather added a smaller but measurable improvement. Third, status classification was 92.3% accurate overall, with errors concentrated between adjacent bands. Fourth, inference on a single record took about 10.4 ms in our environment, which leaves ample margin for an interactive scan-and-response experience. Each of these observations should be read within the stated limits of simulated data.

K. Discussion and Limitations

Several limitations should be kept in mind. First and most important, all quantitative results rest on synthetic data generated from relationships we chose. The experiment confirms that the software chain from tag to prediction works and that the learner can recover a nonlinear degradation law, but it does not establish predictive accuracy on real rails, fastenings or sleepers. Second, the Critical class contains few examples, so rare-event performance is untested; real deployments would need class weighting or targeted data collection. Third, the model estimates a single RUL value without an uncertainty interval, although the spread of the individual trees could be used to produce one.

Fourth, the scheme assumes that telemetry such as axle load and speed can be attached to each asset. In practice these values would be derived from traffic records for the section in which the asset lies, and weather values from the nearest station, so measurement error will enter. Fifth, the laser-marked tag helps identification but does not by itself measure condition; the quality of predictions depends on how often and how consistently readings are collected. Finally, the status thresholds are illustrative and must be agreed with the maintenance authority.

Table 9 Calendar-Based Practice Versus the Proposed Workflow

Aspect	Calendar-based	Smart Track AI
Trigger	Fixed interval	Estimated RUL
Asset ID	Manual entry	QR scan
Traffic/climate	Not reflected	Model inputs
Record keeping	Paper or ledger	Time-stamped log
Decision aid	Inspector judgement	Status band + RUL

Qualitative design comparison; not a measured field result.

Safety and responsibility need explicit attention because the output concerns infrastructure that carries passengers. The platform is intended as decision support, with final authority remaining with qualified railway engineers. False reassurance, where an asset near the end of life is labelled Optimal, is the more dangerous error type, so a deployed system should favour conservative thresholds and report model uncertainty alongside every estimate. Personal data collected by the platform are limited to the identity of the signed-in user and the time of each action, and should be retained only as long as audit requirements demand.

For reproducibility, the synthetic generator, the training and evaluation scripts, and the generated dataset are described in full in the methodology section, and the random seed is fixed so that every table and figure can be regenerated. We make no claim that the dataset resembles real railway records.

Operating the platform over time raises further questions. A model trained on past conditions will drift as traffic patterns, rail grades or climate change, so predictions should be compared with the outcomes of later inspections, and the model retrained on a schedule. Each retrained model should be versioned, and the version used for a prediction should be stored in its record so that results remain traceable. Input checks should flag implausible values, such as a negative age or an axle load far outside the physical range, before they reach the model. Finally, a human override must always be possible: an inspector who sees a cracked weld should be able to mark the asset for action regardless of what the model says.

There are also threats to the validity of the present study. The synthetic data were generated and analysed by the same team, so the learner may implicitly benefit from our knowledge of how the data were built. Hyper-parameters were left near their defaults and the random split was a single draw, although cross-validation partly addresses the latter. The comparison with calendar-based practice in Table IX is a statement about design, not a measurement. We therefore describe our contribution as a validated software pipeline and a documented experimental baseline, to be superseded by evidence from real track.

Despite these limits, the architecture has practical merits. It needs only a camera and a browser on the client, it keeps model logic on the server where it can be updated without

touching field devices, and its audit log supports accountability. These properties make it suitable as a pilot on a limited section before any wider rollout.

CONCLUSION

We presented Smart Track AI, an integrated platform that identifies track assets by laser-marked QR tags, captures service-load and weather data at inspection time, and returns an RUL estimate and healthstatus through a Flask-served Random Forest. On a synthetic evaluation the Random Forest reached RZ 0.945 with an MAE of 5.20 months, and 92.3% of assets were placed in the correct status band; gradient boosting was slightly more accurate. These findings show that the end-to-end design is workable, while the question of real-world accuracy remains open.

Future work will (i) replace the simulator with historical inspection and failure records from a pilot section, (ii) add uncertainty estimates and class-imbalance handling, (iii) test sequence models that use the full inspection history rather than a single snapshot, (iv) evaluate the durability and read rate of laser-marked tags under field conditions, and (v) add an offline mode so that scans can be queued where mobile coverage is poor.

REFERENCES

- [1] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [2] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [3] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. IEEE Int. Conf. Prognostics Health Manage.*, Denver, CO, USA, 2008, pp. 1–9.
- [4] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018.
- [5] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York, NY, USA: Springer, 2009.
- [7] Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification, ISO/IEC Standard 18004, 2015.
- [8] I. Soleimanmeigouni, A. Ahmadi, and A. Arastehkhoh, "Track geometry degradation analysis: A review," *International Journal of Rail Transportation*, vol. 6, no. 3, pp. 137–168, 2018.
- [9] A. R. Andrade and P. F. Teixeira, "Uncertainty in rail track geometry degradation: Information for track maintenance for high-speed lines," *Journal of Transportation Engineering*, vol. 138, no. 11, pp. 1400–1408, 2012.

- [10] A. Lasisi and N. Attoh-Okine, "Principal component analysis and track quality index (TQI) in railway track geometry assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3392–3400, 2018.
- [11] X. S. Si, W. Wang, C. H. Hu, and D. H. Zhou, "Remaining useful life estimation—A review on the statistical data driven approaches," *European Journal of Operational Research*, vol. 213, no. 1, pp. 1–14, 2011.
- [12] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *Proc. IEEE Int. Conf. Prognostics Health Manage. (ICPHM)*, Dallas, TX, USA, 2017, pp. 88–95.
- [13] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, Art. no. 106024, 2019.
- [14] F. Ghofrani, Q. He, R. M. P. Goverde, and X. Liu, "Recent applications of big data analytics in railway transportation systems: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 226–246, 2018.
- [15] Y. Chen, H. Zhang, and X. Liu, "Evaluation and recognition of laser direct part marking (DPM) 2D codes under severe abrasive environments," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 8, pp. 6981–6990, 2020.