Indian Journal of Computer Science and Technology
https://www.doi.org/10.59256/indjcst.20260503123
Volume 5, Issue3 (September-December 2026), PP: 1034-1037.
www.indjcst.com
ISSN No: 2583-5300

Industrial Equipment Failure Prediction System Using Machine Learning

Prasanna kumar Naik R[1], Chethan Kumar G S[2], Swarnalatha[3], Preethi K P[4], Meghana H V[5]

[1,2,3,4]Assistant professor, Department of Master Of Computer Application, University BDT, College Of Engineering (A Constituent College of Vtu,Belagavi) Davangere, Karnataka, India.
[5]Department of Master of Computer Application, University BDT, College of Engineering (A Constituent College Of Vtu,Belagavi), Davangere, Karnataka, India.

To Cite this Article: *Prasanna kumar Naik R[1], Chethan Kumar G S[2], Swarnalatha[3], Preethi K P[4], Meghana H V[5], "Industrial Equipment Failure Prediction System Using Machine Learning", Indian Journal of Computer Science and Technology, Volume 05, Issue 03 (September-December 2026), PP: 1034-1037.*

Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution License; Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: *Unexpected machinery breakdowns in modern manufacturing and heavy industries cause costly unplanned downtime, production losses, and workplace safety hazards. Traditional reactive and fixed-schedule preventive maintenance strategies either respond after failure occurs or replace usable components prematurely, while manual interpretation of multi-sensor readings remains challenging. this paper presents a software-based Industrial Equipment Failure Prediction System utilizing a trained on the AI4I 2020 Predictive Maintenance Dataset. During preprocessing, identifier attributes and failure-mode indicators are excluded to prevent target leakage, leaving five numeric operating parameters and one-hot encoded machine-type features. An 80:20 stratified train test split is applied and class imbalance is mitigated using balanced class weights across a 100 tree Randoms Forest ensemble Evaluated on 2,000 held out test records using default classification output the models achieves 98.10% overall accuracy with a failure class precision of 0.73 recall of 0.69 and F1-score of 0.71 correctly identifying 47 of 68 actual failure cases The serialized model is integrated into both a Streamlit web dashboard and a command-line interface enabling users to inputs sensors readings and obtains Safes or Failures predictions alongside failure risks probabilities to supports proactive maintenances planning.*

Key Word: *AI4I 2020 dataset, class imbalance, equipment failure prediction, machine learning, predictive maintenance, Random Forest, Streamlit.*

I. INTRODUCTION

Modern factory and heavies industries depends on reliables machine to keeps productions steady maintained product qualities and keeps worker safe Where a machines break downs withouts warnings works stop orders are delayed repairs cost rises and workplaces hazards increases With growings automations on there factories floors industrials equipments continuously record operatings dimension such as air diseases developments diseases turnings speeds torques and tool weas Studying there values makes it possibles to notices unusuals patterns early before a majors breakdowns takes place.

Traditionals maintenances strategy usually follows eithers reactives or scheduled preventive methods In reactives maintenances technician repairs a machines only afters stops workings where causeses longs unplanned downtimes and serious productions losses Scheduled preventives maintenances services equipment at fixeded times intervals evens when parts are stills in goods condition resulting in early part replacements and unnecessary expenses Even though modern machinery records multiple sensor values at the same time, checking combinations of these readings manually to spot early signs of failure is slow and easy to get wrong.

Predictive maintenance overcomes these limitations by using actual machine condition data to estimate the chance of failures before a breakdown occurs complex pattern acrosses multiple operating parameters directly from historical machine records. Historical labeled records. Among ensemble learning methods the Random Forest algorithm [2] offers strong predictive capability, robustness against overfitting, and compatibility with tabular industrial datasets.

This paper presents a software-based Industrial Equipment Failure Prediction System built using Python [6] and trained on the AI4I 2020 Predictive Maintenance Dataset [1]. The proposed workflow incorporates structured data preprocessing with Pandas [4]—specifically removing unique identifiers and failure-mode indicator flags to prevent target leakage—and trains a Random Forest classifier using Scikit-learn [3] with balanced class weights to address the inherent scarcity of failure events. The trained classifier and expected feature schema are serialized via Joblib [7] and integrated into both an interactive Streamlit dashboard [5] and a command-line interface (CLI). Through there edges users canned inputs manuals sensors reading to obtain a binary condition classification (*Safe* or *Failure*) along with a model-estimated failure-risk probability. The prototype is designed strictly as a decision-support tool to assist maintenance planning rather than to replace professional engineering inspection or industrial safety protocols.

II. LITERATURE SURVEY

Industrial maintenances methodology had progressively evolved for reactive approaches towards condition based predictives paradigms Under reactive maintenances upkeep are initiated only afters machinery experience functionals breakdowns resulting costly unplanneded downtimes disrupted productions schedules and heightened workplaces safety hazard Scheduleded preventives maintenances mitigates unexpected failures by servicing equipments at predetermined times intervals however becauses fixed intervals do not reflects there actuals degradation states of individual machines serviceables components are frequently replaceded unnecessarily Predictive maintenance bridges there gaps by analyzing operational sensor measurements—such as infection, revolving speed, turning, and tool wear—to estimate failure risk prior to breakdown, enabling timely and cost-effective maintenance planning.

Developing reliable supervised machines learning model require representatives labeled historicals records The AI4I 2020 Predictives Maintenance Dataset hosted by the UCI Machine Learning Repository [1] serves as a widely adopted benchmarks reflecting realistics industrial operating conditions alongside binary machine failures labels (Machine failure) In addition to physicals operating parameters and machine types category there datasets includes specifics failure-mode indicator columns critical methodological consideration when modeling this dataset is preventing target leakage: because these failure-mode flags directly encode the occurrence of failure modes rather than pre-failure sensor states, retaining them in the feature matrix artificially inflates validation metrics while rendering the model unusable for real-time prediction where only sensor readings are available. Removing identifier attributes (UDI, Product ID) and all failure-mode indicators ensures genuine predictive evaluation.

Among supervised classification algorithms, the Random Forest method introduced by Breiman [2] is particularly effective for tabular industrial data. Random Forest constructs an ensemble of decision trees trained on bootstrapped data subsets with randomized feature selection at each split, thereby capturing complex nonlinear interactions among operating variables while reducing variance and overfitting compared to single decision trees. Using the Scikit-learn framework [3], Random Forest classifiers can incorporate cost-sensitive learning via balanced class weighting (class_weight='balanced'), which automatically scales class penalties in reverse relational to class frequencies. This mechanism is essential in trade predictive maintenance, where catastrophe measures constitute a small minority relative to normal operating records, making accuracy alone misleading and necessitating evaluation via failure-class precision, recall, F1-score, and mix-up matrix analysis.

Translating trained models into usable engineering prototypes relies on modular open-source ecosystems built on Python [6]. The Pandas library [4] provides robust structures for tabular ingestion, missing or invalid value handling, numeric type coercion, and one-hot encoding of categorical machine types. Once trained, models and expected feature schemas can be serialized for persistent reuse via Joblib [7], decoupling offline training from inference. Finally, Streamlit [5] enables rapid development of interactive web dashboards directly in Python, allowing practitioners to enter machine parameters and inspect real-time diagnostic classifications and failure probabilities without requiring specialized web-development infrastructure.

While foundational algorithms [2], [3] and benchmark datasets [1] provide the building blocks for predictive maintenance, academic prototypes frequently overlook practical engineering safeguards such as strict removal of failure-mode leakage columns, explicit handling of minority-class recall, schema alignment during inference, and dual accessibility via graphical and command-line interfaces. The proposed system directly addresses these requirements by integrating leakage-free preprocessing, a class-balanced Random Forest model, rigorous minority-class evaluation, and reproducible local deployment.

III. METHODOLOGY

The proposed Industrial Equipment Failure Prediction System follows a structured supervised machine learning workflow implemented in Python [6] using Pandas [4], Scikit-learn [3], Joblib [7], and Streamlit [5].

A. Problem Definition and Dataset Selection

The prediction task is formulated as binary classification using the AI4I 2020 Predictive Maintenance Dataset [1], where Machine failure serves as the target variable (Safe vs. Failure). Five continuous operating measurements—air temperature, process temperature, rotational speed, torque, and tool wear—alongside encoded machine-type indicators are selected as the input attributes.

B. Data Preprocessing and Feature Engineering

The raw CSV dataset is loaded into a tabular structure using Pandas [4]. To construct a valid feature matrix and prevent target leakage, the target column (Machine failure), unique identifiers (UDI, Product ID), and all five failure-mode indicator flags are removed from the input features.

Next, the categorical Type attribute is converted into binary indicator variables via one-hot encoding with the first category dropped, producing two encoded machine-type features. All remaining attributes are coerced to numeric form, missing or invalid entries are handled, and corresponding target labels are retained. The resulting dataset is split into training and testing subsets at an 80:20 ratio with stratification applied on the target variable to preserve class proportions, yielding 2,000 test records.

C. Model Development and Serialization

A RandomForestClassifier [2], [3] is trained on the 80% training partition using 100 decision trees (n_estimators=100) and a fixed random state to guarantee reproducibility. To mitigate the severe class imbalance inherent in industrial failure records, class_weight='balanced' is enabled so that minority failure samples receive proportionally higher

weight during training. Once trained, the fitted classifier and the exact ordered list of input feature names are serialized via Joblib [7] as machine_failure_model.pkl and model_columns.pkl, respectively, ensuring consistent schema alignment during inference.

D. Model Testing and Evaluation

Model performance is evaluated on the held-out 2,000-record test set using the default predict() output. Evaluation computes the confusion matrix, overall accuracy, precision, recall, and F1-score, with primary focus on failure-class recall to assess false-negative risk. During interactive inference, the saved model is loaded to compute failure-risk probabilities for manually entered operating parameters; custom alert probability thresholds (such as 20% or 30%) used in the prediction interfaces are noted as requiring domain validation and consistency prior to operational deployment.

E. Deployment

The trained model is deployed locally through both a Streamlit [5] web dashboard (app.py) and a command-line prediction utility. Both interfaces accept user-entered sensor readings, align input columns against model_columns.pkl, and output a Safe or Failure status along with the predicted failure probability.

Local startup is streamlined using a Windows batch script that navigates to the project directory and invokes the virtual environment's Python interpreter [6] to launch the Streamlit server automatically.

IV.TECHNOLOGY USED

A. Python

Python [6] serves as the foundational programming language for the entire workflow, encompassing data preparation, model training, performance evaluation, the command-line prediction utility, and the web interface (app.py). Execution is isolated within a Python virtual environment to maintain dependency consistency.

B. Pandas

The Pandas library [4] is utilized for tabular data manipulation. It loads the AI4I 2020 CSV dataset, removes non-predictive identifiers (UDI, Product ID) and failure-mode leakage indicators (TWF, HDF, PWF, OSF, RNF), performs one-hot encoding on the categorical Type attribute, coerces features into numeric format, and handles missing or invalid rows.

C. Scikit-learn

Scikit-learn [3] provides the machine learning infrastructure for the system. It performs the stratified 80:20 train-test split, trains the 100-tree RandomForestClassifier [2] with class_weight='balanced' and a fixed random state, and computes the confusion matrix, overall accuracy, precision, recall, and F1-score on the held-out test set.

D. Joblib

Joblib [7] is employed for efficient object persistence. After training, the fitted Random Forest classifier and the exact ordered list of input feature names are serialized as machine_failure_model.pkl and model_columns.pkl, respectively, enabling rapid loading and strict column-order alignment during inference.

E. Streamlit and Command-Line Interface

Streamlit [5] powers the browser-based diagnostic dashboard (app.py), providing input controls for machine type and the five operating parameters (air temperature, process temperature, rotational speed, torque, and tool wear) and displaying the predicted Safe or Failure status alongside the failure-risk probability. A companion command-line interface (CLI) offers lightweight terminal-based testing of sensor readings.

F. Windows Batch Automation

Local deployment is streamlined using a Windows batch file (.bat), which navigates to the project directory and invokes the virtual environment's Python interpreter [6] to launch the Streamlit server (app.py) automatically without manual command entry.

V. RESULTS AND DISCUSSION

A. Experimental Evaluation Results

The trained Random Forest classifier [2], [3] (100 trees, class_weight='balanced') was evaluated on the held-out 20% stratified test partition comprising 2,000 operational records from the AI4I 2020 dataset [1]. All reported evaluation figures correspond to the classifier's default predict() output from the training run. As summarized in Table 1, the model achieved an overall classification accuracy of 98.10%, alongside a minority failure-class precision of 0.73, recall of 0.69, and F1-score of 0.71.

Evaluation Metric	Reported Value
Overall Accuracy	98.10%
Failure Class Precision	0.73
Failure Class Recall	0.69
Failure Class F1-Score	0.71
Test-Set Size	2,000 records

Table 1 Classification Performance on the Held-Out Test Set

Table 2 details there correspondings confusions matrix across there 2000 tests instance consisting of 1932 actual non-failure cases and 68 actuals failures cases There classifiers correctly identified 1915 non failures records (true negatives) and 47 failures record (true positives) while producings 17 falses positives (non-failures flagged as failures) and 21 falses negatives (actual failures misclassified as non-failures).

Actual \ Predicted	Predicted: No Failure	Predicted: Failure
Actual: No Failure	1,915	17
Actual: Failure	21	47

Table 2 Confusion Matrix of the Random Forest Classifier

B. Discussion and Analysis

Although there overall accuracy of 98.10% appears strongest interpreting these metrics in isolations can been misleadings due to there severes classes imbalances failures events accounts for only 68 of the 2000 tests record in industrials predictives maintenances the primary analyticals concerns is there minority failures classes where falses negatives represents undetected equipments faults that may leads to unplanned downtimes equipments damages or workplaces safety hazards.

The failure-class recalls of there approximately 69% indicates while the classes balanced Random Forest successfully detects more two-thirds of actual failures (47 of 68) after excluding target-leaking failure mode columns (TWF, HDF, PWF, OSF, RNF) 21 failures instance remains undetected under there defaults decisions boundary Meanwhile there failures class precisions of 0.73 reflects a moderate false-alarm rate (17 false positives out of 64 total failure predictions) which is operationally acceptables in a maintenances decisions supports settings where preventives inspections is substantially lesses costly then catastrophic breakdown Consequently systems effectiveness should been judged primarily by failure classes recall and the relatives costs of false negatives predictions rather then aggregates accuracies alone.

Furthermore an important distinctions exists betweens offlines evaluations and applications level alertings The metrics reported in Tables II and III reflect the model's default predict() classifications threshold and musts not be represented as performance metrics for there customs 20% or 30% probability alerts thresholds utilized within there separates Streamlit [5] and command line predictions interfaces Lowering there alert thresholds in deployments can improved failures recalls by flagging borderlines risks cases earliers though it simultaneously increases false positive alerts there fore threshold tuning must be systematically validated and standardized prior to operational deployment. Overall, the implemented prototype demonstrates that leakage-free machine learning combined with accessible interactive interfaces provides a practical baseline for data-driven maintenance planning.

VI. CONCLUSION

This paper presented a software-based Industrial Equipment Failure Prediction System designed to support data-driven predictive maintenance using supervised machine learning. Built upon the AI4I 2020 Predictive Maintenance Dataset [1], the system implements a structured workflow comprising leakage-free preprocessing—explicitly removing unique identifiers and failure-mode indicator flags (TWF, HDF, PWF, OSF, RNF)—categorical encoding, stratified 80:20 data partitioning, and training of a 100-tree Random Forest classifier [2], [3] configured with balanced class weights. The serialized model and feature schema [7] are deployed through both an interactive Streamlit dashboard [5] and a command-line utility for real-time manual sensor evaluation.

On the held-out 2,000-record test set using default classification output, the system achieved an overall accuracy of 98.10%, alongside a failure-class precision of 0.73, recall of 0.69, and F1-score of 0.71. Although overall accuracy is high, the failure-class recall of approximately 69% (47 of 68 actual failures detected) demonstrates that missed failures remain a key operational limitation. Furthermore, the current prototype relies exclusively on manually entered parameters rather than live telemetry or automated machinery control, serving strictly as an analytical decision-support tool for maintenance planning.

REFERENCES

1. S. Matzka, “Explainable Artificial Intelligence for Predictive Maintenance Applications,” in Proc. Third International Conference on Artificial Intelligence for Industries (AI4I), 2020.
2. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, “Machine Learning for Predictive Maintenance: A Multiple Classifier Approach,” IEEE Transactions on Industrial Informatics, vol. 11, no. 3,pp. 812–820, 2015.
3. 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, 2019.
4. Breiman, “Random Forests,” Machine Learning, vol. 45, pp. 5–32, 2001.
5. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.
6. He and E. A. Garcia, “Learning from Imbalanced Data,” IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263–1284, 2009.
7. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
8. UCI Machine Learning Repository, “AI4I 2020 Predictive Maintenance Dataset,” 2020. [Online]. Available: https://archive.ics.uci.edu/