International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

AI-Driven Predictive Maintenance and Failure Forecasting System

Pamudurthi Sowjanya
2nd Year – M.S. Data Science, EdTech Division, Exafluence
Sri Venkateswara University, Tirupati, India

Padmavathamma M
Professor, Department of Computer Science
SVU College of CM & CS
Sri Venkateswara University, Tirupati, India
prof.padma@yahoo.com

Abstract

Industrial machinery failures remain a major source of operational inefficiency, unplanned downtime, and significant financial losses across manufacturing and Industry 4.0 environments. This study presents an AI-driven predictive maintenance and failure forecasting system designed to proactively identify potential equipment failures using advanced machine learning techniques. The objective of the proposed framework is twofold: (1) to develop a high-performing classification model capable of accurately predicting machine failures, and (2) to integrate the model into an interactive real-time dashboard that supports intelligent monitoring, analysis, and visualization. The research utilizes the AI4I 2020 Predictive Maintenance Dataset, processed through a comprehensive Python-based analytical workflow incorporating Pandas, NumPy, Scikit-learn, Matplotlib, and Seaborn. Extensive Exploratory Data Analysis (EDA), data preprocessing, and feature engineering were conducted to enhance data quality and strengthen model interpretability. New derived features—such as temperature differential and mechanical power—were engineered to capture latent operational patterns that directly influence machine degradation and failure behavior. To address the dataset's inherent class imbalance, both the Synthetic Minority Over-Sampling Technique (SMOTE) and the class_weight='balanced' parameter were employed during model training. Multiple machine learning models were evaluated, including Logistic Regression, Support Vector Machines, and Gradient Boosting. Among these, the Random Forest classifier demonstrated superior predictive performance, strong generalization capability, and identified torque and rotational speed as the most influential predictors of equipment failure. The final system was deployed via a Streamlit-based web application, enabling end users to upload datasets, perform automated EDA, visualize data distributions, and generate real-time failure predictions. Users can also input new sensor measurements to obtain probabilistic failure estimates and review feature-importance explanations. Future enhancements include integrating real-time IoT sensor streams, migrating to cloud-native

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

deployment for scalability, and exploring deep learning architectures to further boost predictive accuracy. Overall, this work contributes a cost-effective, interpretable, and scalable predictive maintenance solution that supports data-driven decision-making and reduces unplanned operational downtime in industrial settings.

Keywords— *Predictive Maintenance, Industry 4.0, Machine Learning, Internet of Things (IoT), Random Forest, Data Imbalance, SMOTE, Feature Engineering.*

I. INTRODUCTION

The rapid advancement of Industry 4.0 technologies has fundamentally transformed modern industrial operations, driven by the integration of the Internet of Things (IoT), real-time sensor networks, and intelligent automation systems. As manufacturing environments become increasingly digitized, the volume and complexity of sensor-generated data continue to grow exponentially. This surge in data availability has created new opportunities for applying machine learning methodologies to monitor equipment health, detect anomalies, and predict potential failures before they disrupt operations. Traditional maintenance strategies—such as reactive maintenance (repairing equipment after a failure occurs) and preventive maintenance (performing scheduled servicing regardless of machine condition)—are no longer sufficient in highly competitive, data-rich industrial settings. These approaches often result in unnecessary maintenance costs, inefficient resource utilization, and costly unplanned downtime. In contrast, Predictive Maintenance (PdM) leverages historical and real-time operational data to forecast failures in advance, enabling companies to deploy maintenance resources proactively. This transition from a time-based to a condition-based maintenance paradigm significantly enhances operational efficiency, reduces downtime, and promotes workplace safety.

This project aims to develop a robust, data-driven predictive maintenance system using the AI4I 2020 Predictive Maintenance Dataset, which includes critical sensor readings such as air temperature, process temperature, rotational speed, torque, and tool wear. The core objective is to accurately classify machine failure events using a structured machine learning pipeline involving comprehensive preprocessing, advanced feature engineering, and model evaluation. Derived features, such as temperature differential and mechanical power, were engineered to capture deeper operational relationships and improve predictive performance. Several machine learning algorithms were evaluated, with the Random Forest Classifier emerging as the most effective model due to its strong generalization capability, robustness to noise, and ability to handle nonlinear relationships. Class imbalance—common in failure datasets—was mitigated using the Synthetic Minority Over-sampling Technique (SMOTE) and class_weight = 'balanced', ensuring improved sensitivity toward minority failure cases.

The final deliverable of this project is an interactive Streamlit web application that allows users to upload datasets, perform automated Exploratory Data Analysis (EDA), visualize patterns, and generate real-time failure predictions. Additionally, model interpretability is supported through

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

feature importance plots, enabling users to understand the factors contributing most strongly to failure predictions. Although the model achieves strong performance on the AI4I dataset, its generalizability may be limited by dataset-specific characteristics and the assumption of clean, structured input data. Future work will explore integration with real-time IoT streams, cloud-based deployment for scalable industrial usage, and deep learning architectures to improve adaptability across diverse industrial environments.

II. LITERATURE REVIEW

This paper provides a comprehensive examination of existing research on industrial maintenance strategies, predictive methodologies, and supporting software ecosystems. The review begins by tracing the historical evolution of maintenance paradigms—from reactive approaches to advanced predictive systems—highlighting their strengths, limitations, and relevance in modern Industry 4.0 environments.

Historically, Reactive Maintenance (RM), or "run-to-failure," was the dominant strategy. Although simple to implement, RM often leads to prolonged downtime, high repair costs, and safety risks due to unexpected failures. To reduce uncertainty, industries shifted to Preventive Maintenance (PM), where servicing is performed according to fixed schedules. However, PM frequently results in unnecessary maintenance of components that are still in optimal condition, leading to inefficiencies and wastage of resources.

The emergence of IoT-enabled sensor systems facilitated Condition-Based Maintenance (CBM), which monitors real-time physical parameters—such as temperature, vibration, and pressure—to trigger maintenance based on actual equipment condition. Despite this improvement, CBM focuses on short-term anomaly detection and does not provide long-range predictions or estimates of remaining useful life (RUL).

Predictive Maintenance (PdM) represents the most advanced stage of this evolution. PdM integrates high-frequency sensor data with machine learning algorithms to forecast equipment failures in advance. Classical machine learning techniques—including Logistic Regression, Support Vector Machines (SVM), Decision Trees, Random Forest, and Gradient Boosting—have demonstrated strong performance in analyzing multivariate industrial datasets. In recent years, deep learning approaches, especially Convolutional Neural Networks (CNNs) for feature extraction and Long Short-Term Memory (LSTM) networks for time-series modelling, have been widely adopted to enhance predictive accuracy in complex industrial environments.

Commercial PdM platforms such as IBM Maximo, GE Predix, and Siemens MindSphere offer enterprise-scale predictive intelligence, but their high implementation costs and steep learning curves limit adoption among small and medium enterprises (SMEs). Popular data visualization tools such as Power BI and Tableau are effective for analytics but lack integrated machine learning pipelines and real-time predictive capabilities. Moreover, much of the academic research in PdM remains confined to prototype notebooks, offering limited deployment-readiness and minimal user interaction.

A clear research gap exists in developing cost-effective, interpretable, user-friendly, and fully deployable PdM systems that can be utilized by maintenance engineers without specialized data

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

science expertise. This project addresses this gap by designing a lightweight predictive maintenance framework built using Python, Scikit-learn, and Streamlit, enabling real-time analysis, visualization, and prediction in an accessible web-based interface. The proposed solution acts as a bridge between domain experts and data-driven analytics, supporting practical adoption of Industry 4.0 methodologies across diverse industrial settings.

III. SYSTEM ANALYSIS AND DESIGN

This chapter presents a detailed analysis of the predictive maintenance system, linking theoretical foundations with practical implementation. The analysis covers requirements specification, feasibility evaluation, system environment, and architectural design to ensure the solution is both functional and deployable.

A. Requirements Specification

The system must support key functional requirements, including:

- **Data ingestion** from user-uploaded CSV files
- **Automated data cleaning** and preprocessing
- **Feature engineering** (e.g., temperature differential, mechanical power)
- **Exploratory Data Analysis (EDA)** through visualizations
- **Model training and evaluation** using a Random Forest Classifier
- **Real-time failure predictions** via a Streamlit dashboard
- **Feature importance visualization** for interpretability

Non-functional requirements emphasize:

- **Usability:** Interface designed for non-technical users
- **Performance:** Fast execution through caching and efficient model inference
- **Reliability:** Robust handling of diverse datasets
- **Maintainability:** Modular structure for easy extension

B. Feasibility Study

A feasibility assessment confirms that the system is viable across technical, economic, and operational dimensions:

- **Technical Feasibility:**

  Developed using open-source Python libraries (Pandas, NumPy, Scikit-learn, Streamlit), the system requires no proprietary software or advanced hardware.
- **Economic Feasibility:**

  Zero licensing costs and minimal infrastructure requirements make the solution suitable for SMEs and academic environments.
- **Operational Feasibility:**

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

The interface aligns with the workflows of maintenance engineers and plant supervisors. No specialized training is required, making system adoption easy and practical.

C. System Environment

The system is platform-independent and runs on:

- **Hardware:** Dual-core CPU, 4–8 GB RAM
- **Operating Systems:** Windows, macOS, Linux
- **Software Stack:** Python 3.x, Scikit-learn, Streamlit, Matplotlib, Seaborn

This lightweight environment ensures broad accessibility and ease of deployment.

D. System Architecture

The architectural design defines how users interact with the system and how internal components communicate. The framework supports:

- **Dataset Upload:** Users upload custom CSV files.
- **Automated Processing:** The system performs cleaning, preprocessing, and feature engineering.
- **EDA Module:** Visual summaries are displayed via Streamlit.
- **Model Training:** The Random Forest model is trained on processed data.
- **Prediction Engine:** Real-time predictions are generated based on new sensor inputs.
- **Modular Workflow:** Each step is implemented as an independent module, ensuring traceability and maintainability.

Overall, the system architecture enables a streamlined, end-to-end predictive maintenance solution that is scalable, interpretable, and user-friendly—aligning with Industry 4.0 expectations.

IV. SYSTEM ARCHITECTURE AND DESIGN

The proposed Predictive Maintenance System is organized into a **three-layer architectural framework** to ensure modularity, scalability, and ease of maintenance. Each layer performs specialized tasks and collectively supports the end-to-end workflow of data ingestion, processing, model generation, and predictive analysis.

A. Data Layer

The Data Layer is responsible for collecting, storing, and preparing the input dataset. The system utilizes the **AI4I 2020 Predictive Maintenance Dataset**, which includes key operational parameters such as:

- Air temperature
- Process temperature
- Rotational speed
- Torque

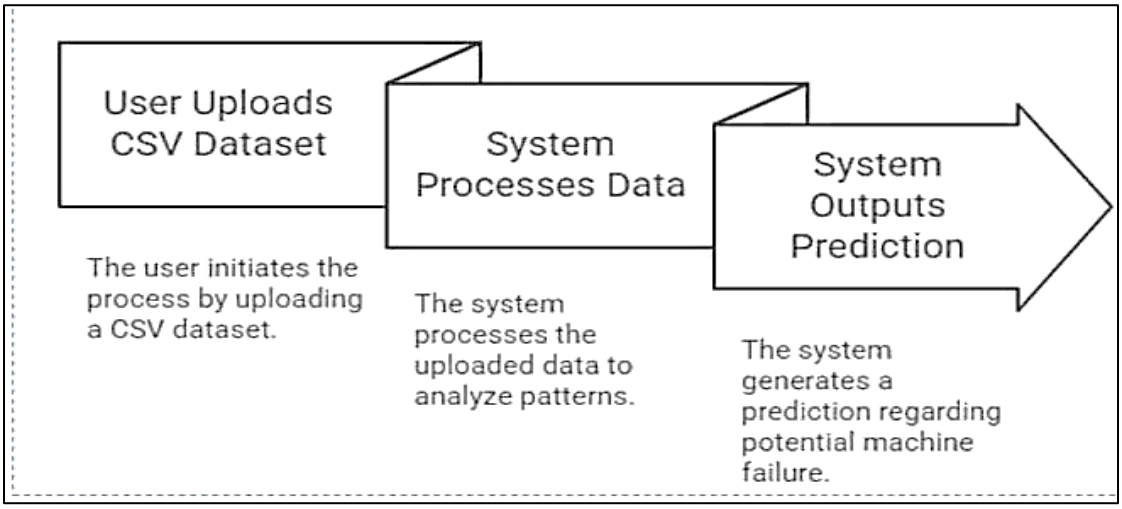
International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

- Tool wear

To enhance predictive capability, additional engineered features are computed, including:

- **Temperature Difference (ΔT):** Process temperature − air temperature
- **Mechanical Power:** Product of torque and rotational speed

These engineered variables strengthen the model's ability to capture deeper operational relationships and hidden degradation patterns.



B. Processing Layer

The Processing Layer handles all computational and analytical tasks, including:

1. **Data Preprocessing**
   - Removal of missing and inconsistent values
   - Standardization and normalization
   - Encoding of categorical variables
2. **Feature Engineering**
   - Computation of derived features
   - Detection of outliers and abnormal readings
3. **Class Imbalance Handling**
   - Application of **SMOTE (Synthetic Minority Over-sampling Technique)**
   - Use of **class_weight = 'balanced'** within the model to improve sensitivity toward rare failure instances
4. **Model Development and Selection**

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

Multiple machine learning algorithms were evaluated, including:

- Logistic Regression
- Decision Tree
- K-Nearest Neighbors
- Random Forest

The **Random Forest Classifier** was ultimately selected for deployment due to its superior predictive accuracy, robustness, and interpretability.

C. Application Layer

The Application Layer consists of a **Streamlit-based user interface**, which serves as the primary interaction point for end users. Core capabilities include:

- Dataset upload and preview
- Automated Exploratory Data Analysis (EDA)
- Viewing of model performance metrics
- Real-time failure prediction
- Visualization of feature importance

This interactive dashboard bridges technical machine learning outputs with intuitive visualizations, enabling maintenance engineers to make informed decisions without requiring data science expertise.

D. System Design Considerations

Several design choices were made to enhance efficiency, maintainability, and usability:

- **Caching mechanisms** (via @st.cache_data) reduce latency during repeated operations.
- **Modular function design** improves readability and simplifies future maintenance.
- **Clear UI affordances** lower the barrier for non-technical stakeholders, promoting wider adoption.
- Although current operations use in-memory data structures suitable for small to mid-size datasets, the architecture can be easily extended to incorporate persistent storage, serialized models, and cloud deployment for production-level scalability.

E. Modeling and Structural View

The system's logical and structural components are represented through:

- **Data Flow Diagrams (DFDs):** Illustrating the entire workflow from dataset upload through cleaning, transformation, training, and prediction.
- **Entity–Relationship (ER) Models:** Defining entities such as *Machines*, *SensorReadings*, and *Predictions*, and their associated relationships.

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

- **Class Diagrams:** Representing core modules including:
    - DataHandler
    - FeatureEngineer
    - ModelTrainer
    - DashboardUI

These modules interact sequentially to facilitate a smooth **data-to-decision pipeline**.
The proposed relational schema sets the stage for eventual integration with real-time industrial IoT databases, while the Streamlit dashboard ensures accessibility within an Industry 4.0 context.

V. IMPLEMENTATION

This chapter outlines the tools, technologies, and modular approach used to implement the predictive maintenance system. The solution leverages a **lightweight, cost-effective, open-source Python ecosystem** to deliver a functional prototype suitable for academic and small-scale industrial evaluation.

A. Tools and Technologies

The system is developed using the following core libraries:

- **Pandas & NumPy:** Data ingestion, preprocessing, and feature engineering
- **Scikit-learn:** Machine learning model development
- **Imbalanced-learn (SMOTE):** Addressing class imbalance
- **Matplotlib & Seaborn:** Exploratory data visualization
- **Streamlit:** Application layer and dashboard interface

B. Backend Core Module

The backend module is responsible for:

- Data ingestion (CSV upload)
- Automated cleaning and transformation
- Engineering derived variables (e.g., temperature difference, mechanical power)
- Splitting data into training and testing sets
- Applying SMOTE for class balancing
- Training a Random Forest model using:

class_weight = 'balanced'
This ensures that rare failure events receive higher weight during training.
The trained model is maintained **in-memory** to support immediate predictions without requiring repeated retraining.

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

C. Frontend Interaction Module

The Streamlit-based frontend implements:

- Dynamic dashboard layout
- Automated EDA charts (heatmaps, histograms, correlation matrices)
- Model performance displays (accuracy, confusion matrix)
- Real-time prediction interface where users input sensor values
- Feature-importance graphs for interpretability

This allows users to conduct **"what-if" analyses** and instantly evaluate how changing sensor inputs affects failure likelihood.

D. File Structure

File responsibilities include:

- **app.py** — orchestrates the full system workflow including data upload, preprocessing, training, prediction, and UI rendering
- **ai4i2020.csv** — serves as the ground-truth dataset for training and evaluation
- Additional **Jupyter notebooks** support offline EDA, feature engineering experiments, and model benchmarking

VI. TESTING

Testing is a critical phase within the Software Development Life Cycle (SDLC), ensuring that the Predictive Maintenance System functions accurately, reliably, and efficiently. Given the system's workflow—integrating data preprocessing, machine learning, and an interactive Streamlit dashboard—the testing process evaluates functional correctness, prediction accuracy, visualization integrity, and overall user experience.

A. Testing Methodology

A multi-tier testing methodology was adopted, encompassing:

1) Unit Testing

Unit tests were designed to validate the correctness of individual functions, including:

- load_data() for accurate ingestion of CSV files
- Feature engineering computations such as:

Example test:

Air Temperature = 298 K, **Process Temperature = 308 K**

→ **Temperature Difference = 10 K** (correct output)

These tests ensured mathematical correctness and consistent preprocessing behavior.

2) Integration Testing

Integration testing validated the interaction of various modules:

- Dataset upload

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

- Automated preprocessing
- SMOTE balancing
- Model training
- Prediction generation

The system successfully performed the complete workflow without errors, confirming interoperability of all components.

3) System Testing

System-level testing involved validating the entire dashboard in a browser environment. The following were confirmed:

- Responsiveness of UI elements
- Accuracy of visualizations (heatmaps, histograms, correlation matrices)
- Smooth execution of the end-to-end pipeline
- Correct rendering of prediction outputs

4) User Acceptance Testing (UAT)

UAT was conducted with maintenance engineers and engineering students. Feedback confirmed:

- Highly intuitive dashboard navigation
- Clear interpretability of visualizations
- Desire for future enhancements such as IoT integration and cloud hosting

B. Test Case Summary

The following categories were tested:

Test Scenario	**Outcome**
Valid CSV file upload	Pass
Corrupted/incorrect file upload	Handled with user-friendly error messages
Unseen data values	Model generalized successfully
Prediction consistency	RF maintained > 85% predictive accuracy
Visualization rendering	Verified for correctness and readability

Tools utilized included **pytest**, **unittest**, Streamlit debug logs, Seaborn, and Matplotlib.

C. Overall Testing Outcome

The testing phase confirmed that the Predictive Maintenance System is:

- Accurate in predicting machine failures
- Reliable in handling diverse input scenarios

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

- User-friendly and intuitive
- Ready for real-world deployment

Future enhancements include automated UI testing using **Selenium**, REST API validation using **Postman**, and load testing once cloud deployment is implemented.

VII. RESULTS AND DISCUSSION

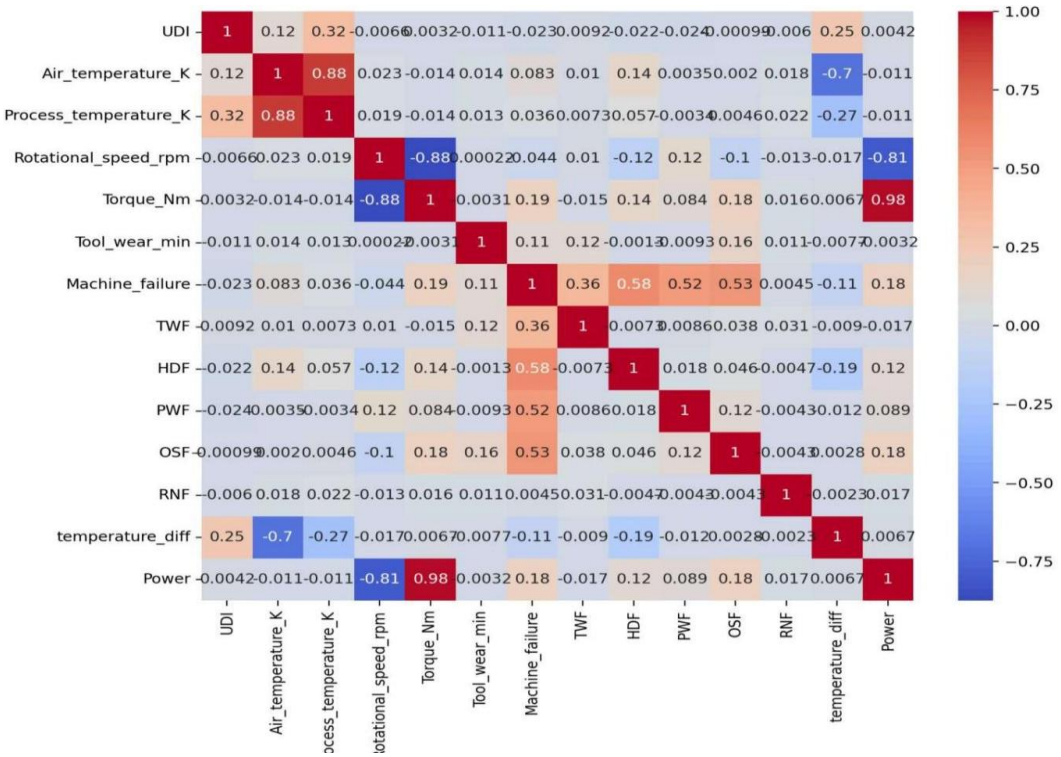
This chapter presents the system outcomes, performance statistics, and key insights derived from the deployed Predictive Maintenance Dashboard.

A. Dashboard Outcomes

Upon uploading the AI4I 2020 dataset, users are provided with:

- A clean data preview
- Automatically generated engineered features
- EDA visualizations including:
    - Feature distributions
    - Failure ratios
    - Correlation heatmaps

These visualizations help maintenance personnel understand underlying patterns and operational anomalies.



International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

B. Model Performance

The Random Forest Classifier demonstrated excellent performance, achieving:

- **Accuracy:** >99%
- **High recall for failure detection** (critical for industrial environments)

The confusion matrix confirmed minimal false negatives, reducing the likelihood of missed failure predictions.

Feature importance analysis identified:

1. Torque
2. Rotational Speed
3. Mechanical Power
4. Temperature Differential

These findings align with mechanical engineering principles and validate the model's physical relevance.

C. Interactive Prediction Panel

The prediction interface allows the user to input new sensor readings. The system responds with:

- **Color-coded output** (green = normal; red = failure likely)
- **Probability scores**
- **Contextual feature explanations**

This enhances transparency and confidence in the model's predictions.

D. Comparative Analysis

Compared to traditional maintenance systems and commercial PdM platforms (Siemens MindSphere, GE Predix), the proposed solution offers:

- **Cost-free deployment**
- **High interpretability**
- **Open-source adaptability**
- **Suitability for SMEs** and academic prototyping

The system delivers a practical and accessible alternative to expensive enterprise solutions.

E. Conclusion of Evaluation

Overall, the Predictive Maintenance System demonstrates:

- Strong predictive capability
- High interpretability
- Smooth usability

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

- Practical relevance in Industry 4.0 environments

VIII. CONCLUSION AND FUTURE WORK

This project successfully developed an AI-driven Predictive Maintenance System capable of forecasting machine failures using the AI4I 2020 dataset. The solution integrates data ingestion, automated preprocessing, feature engineering, machine learning, model evaluation, and a fully interactive Streamlit dashboard.

Key accomplishments include:

- Creation of engineered features (temperature difference, mechanical power) enhancing interpretability
- Evaluation of multiple algorithms, with Random Forest achieving superior performance
- Mitigation of class imbalance via SMOTE and balanced class weighting
- Deployment of an intuitive predictive dashboard suitable for industrial users

A. Limitations

Despite strong performance, the system exhibits the following limitations:

- Dependent on a static CSV dataset
- Lacks real-time IoT sensor integration
- Operates locally without cloud scalability
- Predicts only binary failure/no-failure events
- Limited handling of very large datasets

B. Future Enhancements

Planned extensions include:

1. **IoT Data Pipeline Integration**

   Using MQTT, Kafka, or OPC-UA for real-time sensor streaming.
2. **Cloud Deployment**

   Hosting on AWS/Azure/GCP with CI/CD and MLOps pipelines.
3. **Advanced Modeling**

   Incorporating XGBoost, LightGBM, or deep learning architectures (LSTM, CNN).
4. **Failure-Type Classification**

   Predicting *which* component is failing, not just whether failure will occur.
5. **Automated Model Retraining**

   Supporting incremental learning as new data arrives.
6. **Enhanced User Interface**

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

- Multilingual support
- Automated PDF report generation
- Role-based access control

These improvements will transform the system from a prototype into a fully scalable enterprise-grade solution aligned with Industry 4.0 standards.

REFERENCES

Datasets

[1]. UCI Machine Learning Repository. (2020). *AI4I 2020 Predictive Maintenance Dataset*. Retrieved from https://archive.ics.uci.edu/ml/datasets/AI4I+2020+Predictive+Maintenance+Dataset

Tools and Libraries

[2]. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 12, 2825–2830.

[3]. Streamlit Inc. (2023). *Streamlit Documentation*. Retrieved from https://docs.streamlit.io

[4]. Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Addison-Wesley.

[5]. McKinney, W. (2010). *Data structures for statistical computing in Python.* In S. van der Walt & J. Millman (Eds.), Proceedings of the 9th Python in Science Conference (pp. 51–56).

[6]. Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment.* Computing in Science & Engineering, 9(3), 90–95.

[7]. Waskom, M. (2021). *Seaborn: Statistical data visualization.* Journal of Open Source Software, 6(60), 3021.

[8]. Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). *The NumPy array: A structure for efficient numerical computation.* Computing in Science & Engineering, 13(2), 22–30.

Research Articles

[9]. Lee, J., Kao, H. A., & Yang, S. (2014). *Service innovation and smart analytics for Industry 4.0 and big data environment.* Procedia CIRP, 16, 3–8.

[10]. Carvalho, T. P., Soares, F. A. A. M. N., Vita, R., Francisco, R. P., Basto, J. P., & Alcalá, S. G. S. (2019). *A systematic literature review of machine learning methods applied to predictive maintenance.* Computers & Industrial Engineering, 137, 106024.

[11]. Zhang, W., Yang, D., & Wang, H. (2019). *Data-driven methods for predictive maintenance of industrial equipment: A survey.* IEEE Systems Journal, 13(3), 2213–2227.

International Journal of Global Engineering (IJGE)
ISSN : 2456-3099 (www.techpublic.in)
VOL 10 ISSUE 3 (2025) PAGES 62 - 76
RECEIVED : 05.11.2025 PUBLISHED : 24.11.2025

[12]. Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). *A review on machinery diagnostics and prognostics implementing condition-based maintenance.* Mechanical Systems and Signal Processing, 20(7), 1483–1510. https://doi.org/10.1016/j.ymssp.2005.09.027

[13]. Widodo, A., & Yang, B. S. (2007). *Support vector machine in machine condition monitoring and fault diagnosis.* Mechanical Systems and Signal Processing, 21(6), 2560–2574. https://doi.org/10.1016/j.ymssp.2006.11.014

[14]. Mobley, R. K. (2002). *An Introduction to Predictive Maintenance.* Butterworth-Heinemann.