

CAPSTONE PROJECT

PREDICTIVE MAINTENANCE MODEL

Presented By:

1. Kasetti Harish-Gokaraju Rangaraju Institute Of Engineering-CSE

OUTLINE

- Problem Statement
- Proposed System/Solution
- System Development Approach
- Algorithm & Deployment
- Result (Output Image)
- Conclusion
- Future Scope
- References

PROBLEM STATEMENT

Unexpected failures in industrial machinery cause significant downtime, high repair costs, and production delays. Traditional maintenance strategies like reactive or time-based servicing often miss early signs of failure, leading to inefficient resource use. With modern machines generating vast sensor data, manually detecting failure patterns is difficult and error-prone.

There is a critical need for a system that can automatically analyze this data and classify potential failure types in advance to improve maintenance planning and operational efficiency. Accurate early detection not only reduces unplanned outages but also extends equipment lifespan. A smart data-driven approach can help industries shift from reactive to proactive maintenance strategies.

PROPOSED SOLUTION

- This project aims to build a machine learning-based predictive maintenance model that can accurately classify the type of failure (e.g., tool wear, heat dissipation failure, power failure, etc.) using real-time and historical sensor data.
- Data Collection:
 - Sensor readings: air temperature, process temperature, torque, rotational speed, etc.
 - Labeled failure types: e.g., Heat Dissipation Failure, Tool Wear Failure, Power Failure
- Data Preprocessing:
 - Cleaned data by removing nulls, handling outliers, and converting categorical variables.
 - Split data into training and testing sets for model evaluation.
- Machine Learning Algorithm:
 - Automatically test multiple classifiers (Random Forest, XGBoost, etc.)
 - Perform feature selection and hyperparameter tuning
 - Select the best pipeline using accuracy, F1-score, and confusion matrix
- Deployment:
 - Deployed the selected model as an online endpoint using IBM Watson Machine Learning.
 - Allows real-time prediction of failure type from new sensor inputs via REST API.
- Evaluation:
 - Accuracy, Precision, Recall, F1-score
 - Confusion Matrix to analyze class-level performance

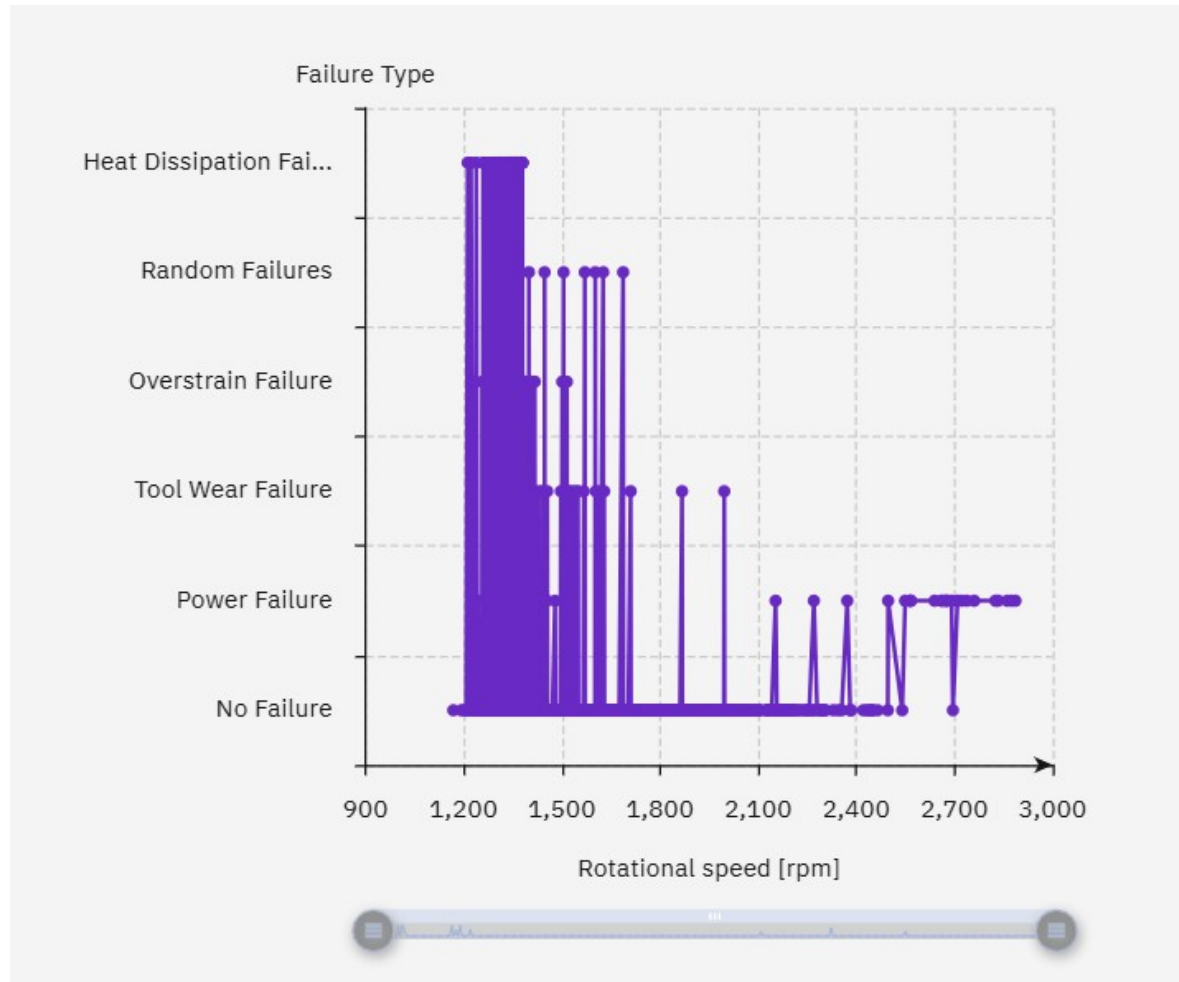
SYSTEM APPROACH

- System requirements:
 - i IBM Cloud Lite Account (Free tier)
 - ii Watson Studio Project
 - iii Watsonx.ai (for model training and AutoAI)
 - iv Cloud Object Storage (for dataset storage)
 - v Watson Machine Learning (for deployment)
- Library required to build the model
 - i Pandas – Data loading, cleaning, and manipulation
 - ii Matplotlib, seaborn – Data visualization
 - iii scikit-learn –Preprocessing, training classifiers, evaluation metrics
 - iv numpy –Numerical operations

ALGORITHM & DEPLOYMENT

- **Algorithm Selection:**
 - Random Forest Classifier – chosen for its accuracy, robustness, and ability to handle complex sensor data.
 - Handles high-dimensional sensor data, Robust to noise & outliers
 - Provides interpretable feature importance
 - High accuracy for classification problems (e.g., tool wear, power failure)
- **Data Input:**
 - Sensor Features
 - Air temperature, process temperature
 - Rotational speed, torque, Tool wear
- **Training Process:**
 - Data cleaned & split.
 - Trained using IBM Watsonx.ai & AutoAI with tuning and validation.
- **Prediction Process:**
 - Real-time sensor input predicts failure type.
 - Deployed via IBM Cloud – enabling early maintenance & reducing downtime.

RESULT



RESULT

IBM watsonx.ai Studio

Search in your workspaces

Upgrade ?

Harish Kasetti's Account

Sydney

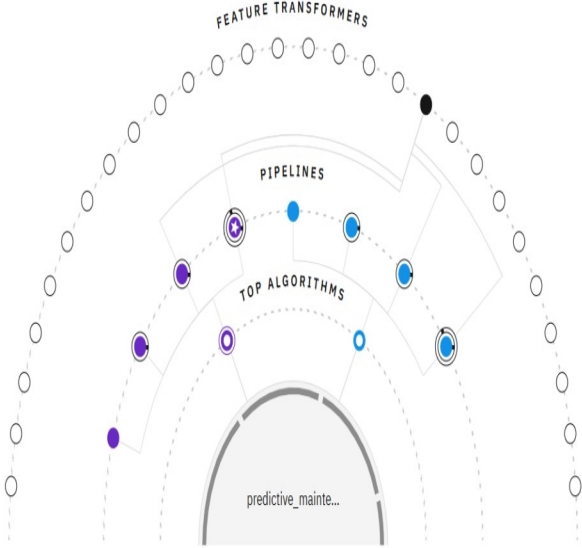
HK

Projects / NSAP_PROJECT / NSAP_MODEL

Experiment summary Pipeline comparison

★ Rank by: Accuracy (Optimized) | Cross validation score

Relationship map ⓘ
Prediction column: Failure Type



Progress map
Swap view ↗

Experiment completed ✓
8 PIPELINES GENERATED
8 pipelines generated from algorithms. See pipeline leaderboard below for more detail.
Time elapsed: 3 minutes

View log Save code

Pipeline leaderboard ▾

Rank	↑	Name	Algorithm	Specialization	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★ 1		Pipeline 4	○ Snap Random Forest Classifier		0.981	HPO-1 FE HPO-2	00:00:46
2		Pipeline 3	○ Snap Random Forest Classifier		0.981	HPO-1 FE	00:00:38
3		Pipeline 2	○ Snap Random Forest Classifier		0.977	HPO-1	00:00:16
4		Pipeline 1	○ Snap Random Forest Classifier		0.977	None	00:00:03

IBM watsonx.ai Studio

Search in your workspaces

Upgrade ?

Harish Kasetti's Account

Sydney

HK

Projects / NSAP_PROJECT / NSAP_MODEL

Experiment summary Pipeline comparison

★ Rank by: Accuracy (Optimized) | Cross validation score



Time elapsed: 3 minutes

View log Save code

Pipeline leaderboard ▾

Rank	↑	Name	Algorithm	Specialization	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★ 1		Pipeline 4	○ Snap Random Forest Classifier		0.981	HPO-1 FE HPO-2	00:00:46
2		Pipeline 3	○ Snap Random Forest Classifier		0.981	HPO-1 FE	00:00:38
3		Pipeline 2	○ Snap Random Forest Classifier		0.977	HPO-1	00:00:16
4		Pipeline 1	○ Snap Random Forest Classifier		0.977	None	00:00:03

RESULT

Prediction results



Prediction type

Multiclass classification

Prediction percentage

3 records

No Failure

Power Failure

Display format for prediction results

☒ Table view

☐ JSON view

Show input data

	Prediction	Confidence
1	No Failure	100%
2	Power Failure	100%
3	No Failure	80%
4		
5		
6		
7		
8		
9		
10		
11		

Download JSON file

RESULT

IBM watsonx.ai Studio

Search in your workspaces

Upgrade

1

Harish Kasetti's Account

Sydney

HK

Deployment spaces / Deploy_1 / P4 - Snap Random Forest Classifier: NSAP_MODEL /

P4 - Snap Random Forest Classifier: NSAP_MODEL

NSAP_MLMODEL Deployed Online

API reference

Test

Public endpoint

<https://au-syd.ml.cloud.ibm.com/ml/v4/deployments/c5733be1-c52b-41f5-bf36-d9be0bf0cd9e/predictions?version=2021-05-01>

[Learn more](#) about the 2021-05-01 version query parameter

Code snippets

cURL	Java	JavaScript	Python	Scala
------	------	------------	--------	-------

```
import java.io.*;
import java.net.MalformedURLException;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
public class HttpClientTest {
    public static void main(String[] args) throws IOException {

        // NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account. (https://au-syd.dai.cloud.ibm.com/docs/content/wsj/analyze-data/ml

        String API_KEY = "<your API key>";
```

https://au-syd.dai.cloud.ibm.com/ml-runtime/models/79ca7faf-ed23-405c-9879-ffd84535b556?space_id=5f38726f-2df3-4325-a8a5-0387de465029&context=cpdaas

Show more

CONCLUSION

- The developed predictive maintenance model using the Random Forest Classifier successfully identified key failure types such as tool wear, heat dissipation, and power failure. Leveraging IBM Watsonx.ai and AutoAI enabled efficient model training and deployment, supporting proactive maintenance and reducing machine downtime. While the system performed well, challenges like class imbalance and limited real-time data integration were encountered. Future enhancements could include advanced models (e.g., XGBoost, LSTM) and real-time monitoring features. Accurate failure prediction is essential to optimize maintenance schedules, reduce operational costs, and improve industrial safety.

FUTURE SCOPE

- To further improve the system, additional data sources such as real-time sensor feeds, environmental conditions, and maintenance logs can be integrated to enhance prediction accuracy. Optimizing the algorithm through hyperparameter tuning or exploring advanced models like XGBoost, LSTM, or ensemble techniques could yield better performance. The solution can also be scaled to monitor machinery across multiple industrial sites or geographic regions, enabling centralized maintenance management. Integration of emerging technologies like edge computing can allow real-time, on-device failure detection, reducing latency and enhancing responsiveness. Incorporating IoT platforms and predictive dashboards will further strengthen decision-making and operational efficiency.

REFERENCES

- Kaggle Dataset – Machine Predictive Maintenance Classification

<https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification>

- IBM Watsonx.ai Documentation – IBM AI and ML services

<https://www.ibm.com/docs/en/watsonx>

- IBM AutoAI Overview – Automated model development by IBM

<https://www.ibm.com/cloud/watson-studio/autoai>

- Edge Computing in Predictive Maintenance – IEEE Xplore

K. Lee, et al., "Edge Computing for Smart Manufacturing," IEEE Access, 2020.

- Scikit-learn Documentation – For Random Forest Classifier

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

- Research Paper – A Review on Predictive Maintenance Techniques

C. Carvalho et al., Journal of Industrial Information Integration, 2019.

IBM CERTIFICATIONS

In recognition of the commitment to achieve
professional excellence



Harish Kasetti

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence



Issued on: Jul 19, 2025
Issued by: IBM SkillsBuild

Verify: <https://www.credly.com/badges/8c6b109f-621d-437c-8f25-6265de516207>



IBM CERTIFICATIONS



IBM CERTIFICATIONS

IBM **SkillsBuild**

Completion Certificate



This certificate is presented to

Harish Kasetti

for the completion of

**Lab: Retrieval Augmented Generation with
LangChain**

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 24 Jul 2025 (GMT)

Learning hours: 20 mins



THANK YOU