# MACHINE LEARNING PROJECT

## PREDICTIVE MAINTENANCE OF INDUSTRIAL MACHINERY

**Presented By:**
**Subhradip Halder – Institute of Engineering and Management, Kolkata – BCA**

edunet foundation

# CONTENTS


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

# PROBLEM STATEMENT

Develop a predictive maintenance model for a fleet of industrial machines to anticipate failures before they occur. This project will involve analyzing sensor data from machinery to identify patterns that precede a failure. The goal is to create a classification model that can predict the type of failure (e.g., tool wear, heat dissipation, power failure) based on real-time operational data. This will enable proactive maintenance, reducing downtime and operational costs.

# PROPOSED SOLUTION

So, our big idea is to build a super smart system that can guess when those machines are gonna act up. We're talking about a "predictive maintenance" model! It'll use machine learning to peek at all the data from machine sensors and spot liflle clues that a breakdown is coming. Here's how we're gonna do it:

**Data Collection:**

- First off, we'll grab all the historical and real-time info from the machine sensors – things like temperature, how much it's shaking, pressure, speed, and even how worn out the tools are.
- We're even using this cool dataset from Kaggle: hflps://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-clas sification

**Data Preprocessing:**

- Raw data can be messy, so we'll clean it all up! No missing bits, no weird numbers, just neat data.
- Then, we'll get clever and create new, helpful "features" from that data – stuff that really helps us predict different kinds of failures.

**Machine Learning Algorithm:**

- This is where the magic happens! We'll pick a smart classification model that can tell us *what kind* of failure is coming. Is it a worn-out tool? Too hot? Power going out? Or is everything just fine?
- We're thinking about using cool algorithms like Support Vector Machines (SVM), Random Forests, or maybe even some fancy Neural Networks.
- We'll teach this model using all that historical sensor data, telling it exactly what kind of failure happened when.

**Deployment:**

- Once our model is trained, we'll set up a way for it to slurp up all that real-time data from the machines.
- We're gonna use IBM Cloud Lite services to run our smart model! Super easy for real-time predictions.

**Evaluation:**

- We'll check how good our model is at guessing! We'll use special scores like accuracy, precision, and recall to see if it's hifling the mark, especially since there are different kinds of failures.
- We'll also do some checks to make sure our model works well on new data it hasn't seen before.
- And we'll keep an eye on it once it's running, making sure it stays sharp and accurate!

# SYSTEM APPROACH

We're tackling this project step-by-step, just like any good machine learning adventure!

**System Requirements:**

- **Hardware:** We'll need enough computer power to crunch all that data and train our model.
- **Software:** We'll be using Python, along with awesome libraries like Pandas, NumPy, Scikit-learn, and maybe TensorFlow/Keras if we get into deep learning.
- **Cloud Platform:** IBM Cloud Lite services are mandatory for getting our model up and running and maybe even storing some data.

**Libraries Required to Build the Model:**

- **For Data:** pandas and numpy – they're like our data wranglers!
- **For Machine Learning:** scikit-learn , xgboost or lightgbm and tensorflow or keras (if we're diving into the deep end with neural networks!).
- **For Pretty Pictures:** matplotlib and seaborn – gotta visualize those results!

# ALGORITHM & DEPLOYMENT

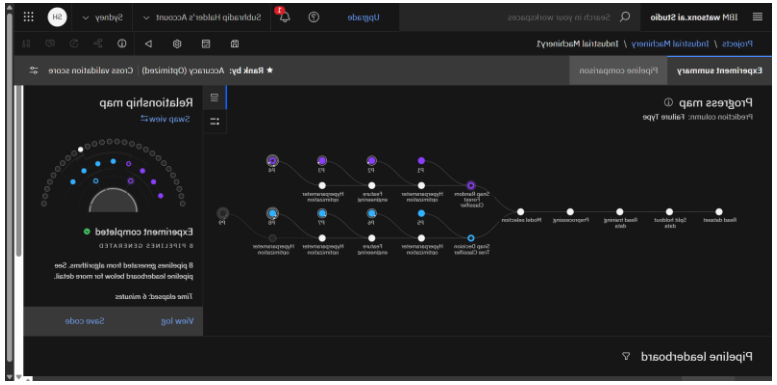
**Algorithm Selection:**
We're gonna try out a few different smart guessing methods and see which one works best:
•**Random Forest:** This one's pretty tough, handles lots of different data, and even tells us what info was most important.
•**Gradient Boosting (XGBoost/LightGBM):** These are often super accurate and fast!
•**Support Vector Machine (SVM):** Great for separating different types of data.
•We'll pick the winner based on how accurate it is, how easy it is to understand, and how fast it runs on our data.

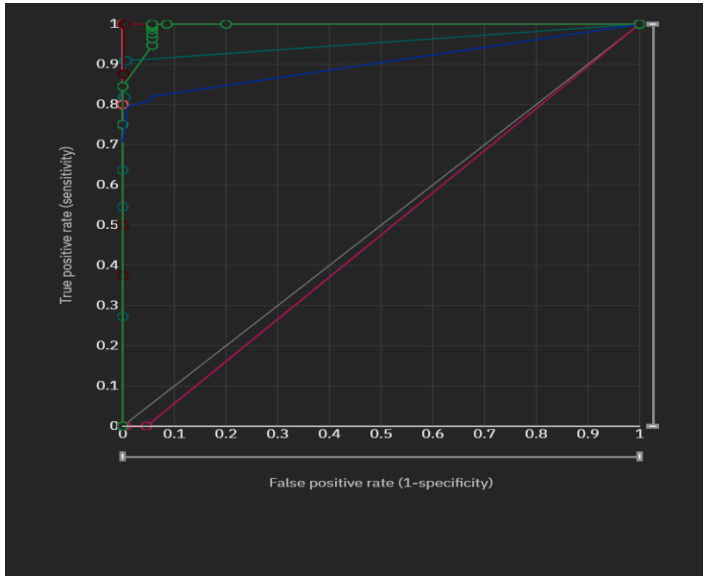
**Data Input:**
Our model will take in all that cleaned-up sensor data, including things like:
•Air temperature
•Process temperature
•How fast it's spinning
•How much force it's putting out
•How much the tool is worn down
•And the machine's quality type

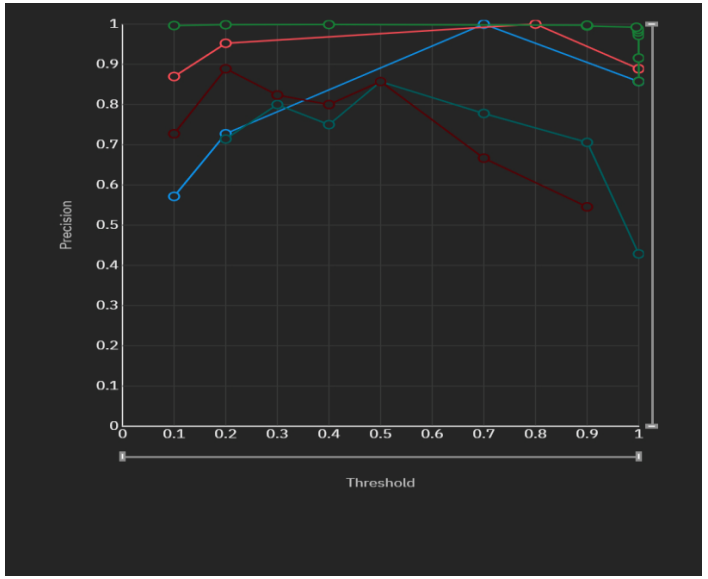
**Training Process:**
•We'll teach our chosen algorithm using all the historical sensor data from that Kaggle dataset.
•We'll split the data up – some for teaching (like 80%) and some for testing (like 20%).
•We'll use cool tricks like "cross-validation" to make sure our model isn't just memorizing but actually *learning*.
•And we'll fine-tune it to make sure it's performing its absolute best!

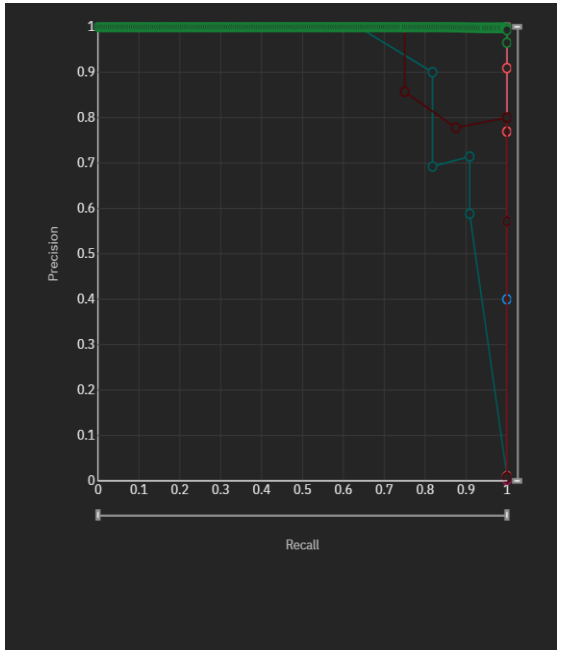
**Prediction Process:**
•Once it's all trained up, our model will get new, live data from the machines.
•We'll clean that new data up just like we did for training.
•Then, BAM! The model will tell us if a failure is likely and what kind it might be. "Tool wear coming up!" or "Looks like a power issue!" or "All clear!"
•We'll set up some "alarm bells" so maintenance teams get a quick heads-up!a

# RESULT

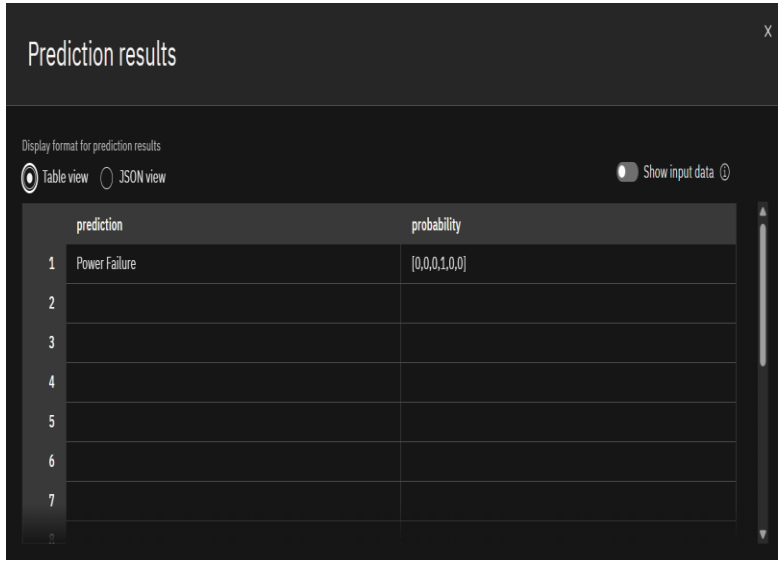





| Observed | Predicted | | | | | | |
|---|---|---|---|---|---|---|---|
| | Heat Dissipation Failure | No Failure | Overstrain Failure | Power Failure | Random Failures | Tool Wear Failure | Percent correct |
| Heat Dissipation Failure | 10 | 0 | 1 | 0 | 0 | 0 | 90.9% |
| No Failure | 0 | 965 | 0 | 0 | 0 | 0 | 100.0% |
| Overstrain Failure | 0 | 0 | 8 | 0 | 0 | 0 | 100.0% |
| Power Failure | 0 | 0 | 0 | 10 | 0 | 0 | 100.0% |
| Random Failures | 0 | 2 | 0 | 0 | 0 | 0 | 0.0% |
| Tool Wear Failure | 0 | 0 | 0 | 0 | 0 | 4 | 100.0% |
| Percent correct | 100.0% | 99.8% | 88.9% | 100.0% | 0.0% | 100.0% | 99.7% |




edunet foundation

# CONCLUSION

So, to wrap it up, we built a predictive maintenance model that can see machine failures coming! By looking at all that sensor data, our smart model can tell us what's about to go wrong, which means maintenance can jump in before things totally stop. This saves a ton of headaches, keeps costs down, and makes our machines way more reliable.

- Sure, we hit a few bumps – like dealing with data where some failures were super rare, or making sure the model worked perfectly on all kinds of machines. But hey, that's part of the fun! For the future, we could add even more data, try out even cooler deep learning stuff, and connect it right into existing maintenance systems. Getting these predictions right is super important for keeping industrial operations smooth and productive!

# FUTURE SCOPE

What's next for this awesome project? So many possibilities!

- **IoT Integration:** Imagine connecting it directly to all those smart sensors for instant data and alerts!
- **Spotting the Weird Stuff:** We could teach it to find totally new, unexpected problems too.
- **Smart Maintenance Schedules:** Maybe even use AI to figure out the *best* time to do maintenance, not just when something's about to break.
- **On-the-Spot Predictions:** Making the model smaller so it can run right on the machine itself for super-fast guesses!
- **Digital Twins:** Building virtual copies of machines to predict their behavior even better.
- **Go Big!** Scaling this up to handle tons of machines across different factories or even cities!

# REFERENCES


- The awesome Kaggle Dataset: "Predictive Maintenance Dataset" by Shivam Bansal. https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification
- All those smart papers about predictive maintenance, sensor data, and machine learning for industry.
- The docs for all the Python libraries we used (Pandas, Scikit-learn, etc.).
- And, of course, the IBM Cloud documentation – super helpful!

# IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Getting Started with Artificial Intelligence

IBM SkillsBuild

IBM

Subhradip Halder

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence

Issued on: Jul 15, 2025

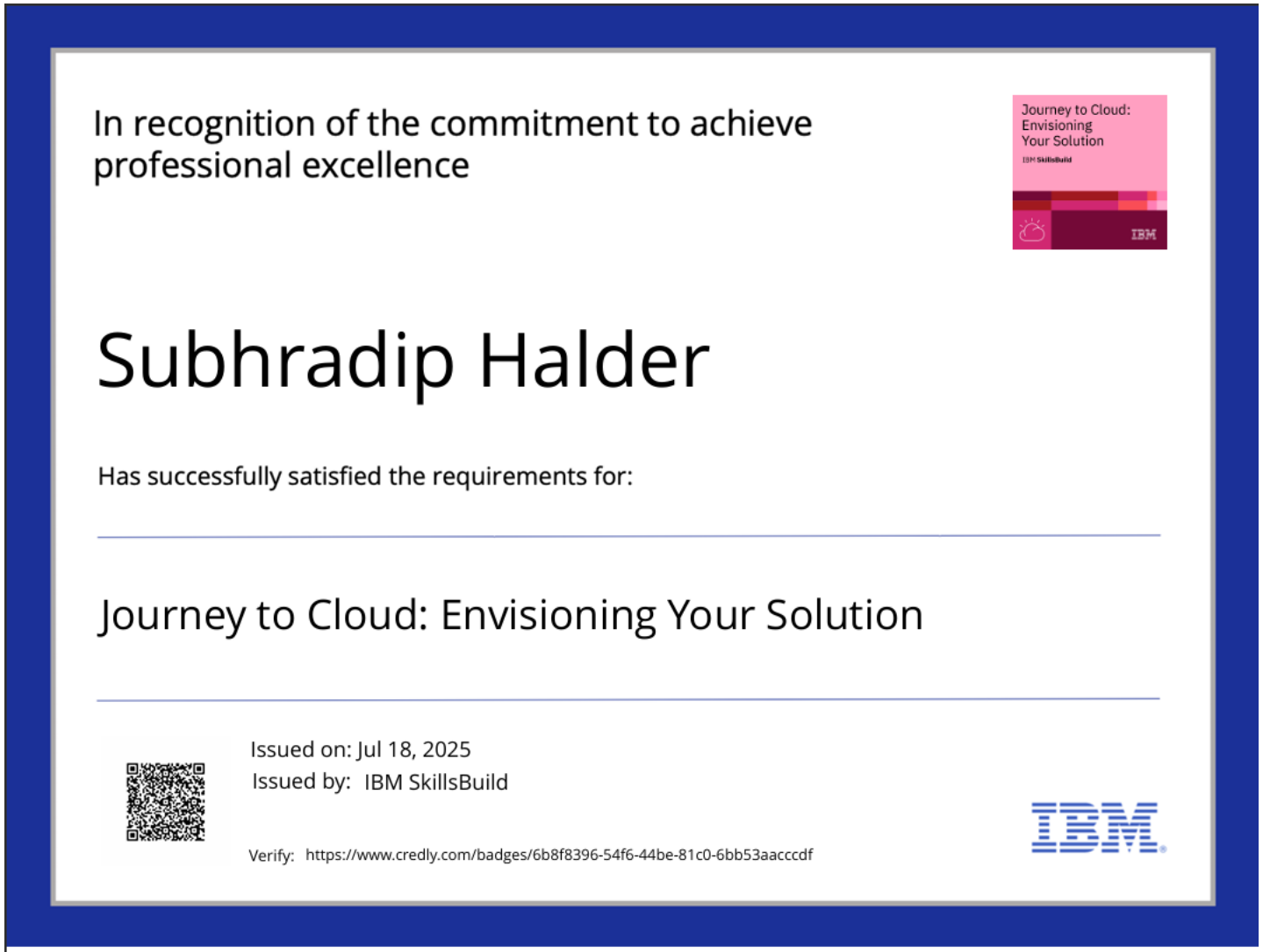
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/b7b1dbb3-626e-4610-b7c6-002c750d8a27

IBM

edunet foundation

# IBM CERTIFICATIONS



In recognition of the commitment to achieve professional excellence

Journey to Cloud: Envisioning Your Solution
IBM SkillsBuild

IBM

Subhradip Halder

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

Issued on: Jul 18, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/6b8f8396-54f6-44be-81c0-6bb53aacccdf

IBM

# IBM CERTIFICATIONS

IBM **SkillsBuild** Completion Certificate

This certificate is presented to

Subhradip Halder

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

GitHub Repo: https://github.com/SubhradipH/EduNet_Internship

# THANK YOU

edunet foundation