Predictive Maintenance for Engine Condition

Catching Failures Before They Happen

Interim Project Report

Ramnathan Ravindran

Capstone Project

January 2026

The Cost of Failure

Every unexpected engine failure costs thousands in repairs, hours of downtime, and puts safety at risk.

The Business Problem

Engine failures don't announce themselves. One moment everything seems normal; the next, a vehicle is stranded, a generator stops mid-operation, or a piece of machinery grinds to a halt.

For businesses that rely on engines - fleet managers, equipment rental companies, manufacturers - these failures create a cascade of problems:

- **Expensive emergency repairs**: Unplanned breakdowns typically cost significantly more than scheduled maintenance
- **Operational downtime**: Every hour of downtime means lost productivity and revenue
- **Safety risks**: Engine failures in critical situations can endanger operators and bystanders
- **Reputation damage**: Unreliable equipment erodes customer trust

The Traditional Approach Falls Short

Most organizations rely on **time-based maintenance** - servicing equipment every X months or Y miles, regardless of actual condition. This approach has two fundamental problems:

1. **Over-maintenance**: Healthy engines get serviced unnecessarily, wasting time and money
2. **Under-maintenance**: Failing engines slip through because they haven't hit the arbitrary threshold

The question isn't "when was the last service?"

The question is "what is the engine telling us right now?"

Modern engines generate continuous streams of sensor data - temperature, pressure, RPM - that contain early warning signals. The challenge is knowing how to read them.

The Opportunity

What If We Could Predict Failures?

Imagine receiving an alert: "Engine #47 showing early signs of trouble. Schedule inspection within 2 weeks."

No emergency. No breakdown. No scrambling for parts. Just a calm, planned intervention that costs a fraction of an emergency repair.

This is **predictive maintenance** - using data to identify problems before they become failures.

Our Objective

Build an intelligent system that:

1. **Analyzes engine sensor data** in real-time
2. **Identifies patterns** that precede failures
3. **Classifies engine condition** as Normal or Needs Attention
4. **Enables proactive intervention** before breakdown occurs

The Business Value

Benefit	Impact
Reduced unplanned downtime	Catch 97% of issues before they cause failures
Lower repair costs	Planned maintenance is more cost-effective than emergency repairs
Extended equipment life	Early intervention prevents cascading damage
Improved safety	Fewer in-field failures means fewer dangerous situations
Data-driven decisions	Move from "gut feel" to evidence-based maintenance

Target Applications

This solution applies to any engine-powered equipment:

- **Vehicle fleets**: Delivery trucks, service vehicles, rental cars
- **Outdoor power equipment**: Commercial lawnmowers, generators
- **Industrial machinery**: Compressors, pumps, manufacturing equipment
- **Marine & aviation**: Boats, small aircraft engines

Our Data Asset (Data Registration)

The Dataset

We analyzed **19,535 engine sensor readings** capturing six key measurements that engines continuously report:

Sensor	Unit	What It Tells Us
Engine RPM	RPM	How hard the engine is working
Lubricating Oil Pressure	bar	Health of the lubrication system
Fuel Pressure	bar	Fuel delivery system status
Coolant Pressure	bar	Cooling system integrity
Lubricating Oil Temperature	°C	Engine heat via oil
Coolant Temperature	°C	Engine heat via coolant

Each reading is labeled with the actual engine condition at that time: **Normal** or **Faulty**.

Data Quality

- **No missing values**: Complete sensor coverage across all readings
- **No duplicate records**: Each observation is unique
- **Labeled outcomes**: Engine conditions recorded for supervised learning

Production-Ready Pipeline

To ensure reproducibility and enable automation, we registered our data assets on Hugging Face Hub:

Resource	Location
Dataset	https://huggingface.co/datasets/spac1ngcat/cs-pred-maintain-ds
Trained Model	https://huggingface.co/spac1ngcat/cs-pred-maintain-model

This allows any downstream system to pull the latest data and model programmatically - essential for production deployment.

What the Sensors Tell Us (Exploratory Data Analysis)

Discovery 1: Engine RPM Is the Strongest Signal

When we examined what distinguishes healthy engines from failing ones, **Engine RPM emerged as the most predictive feature** - though not in the way one might expect.

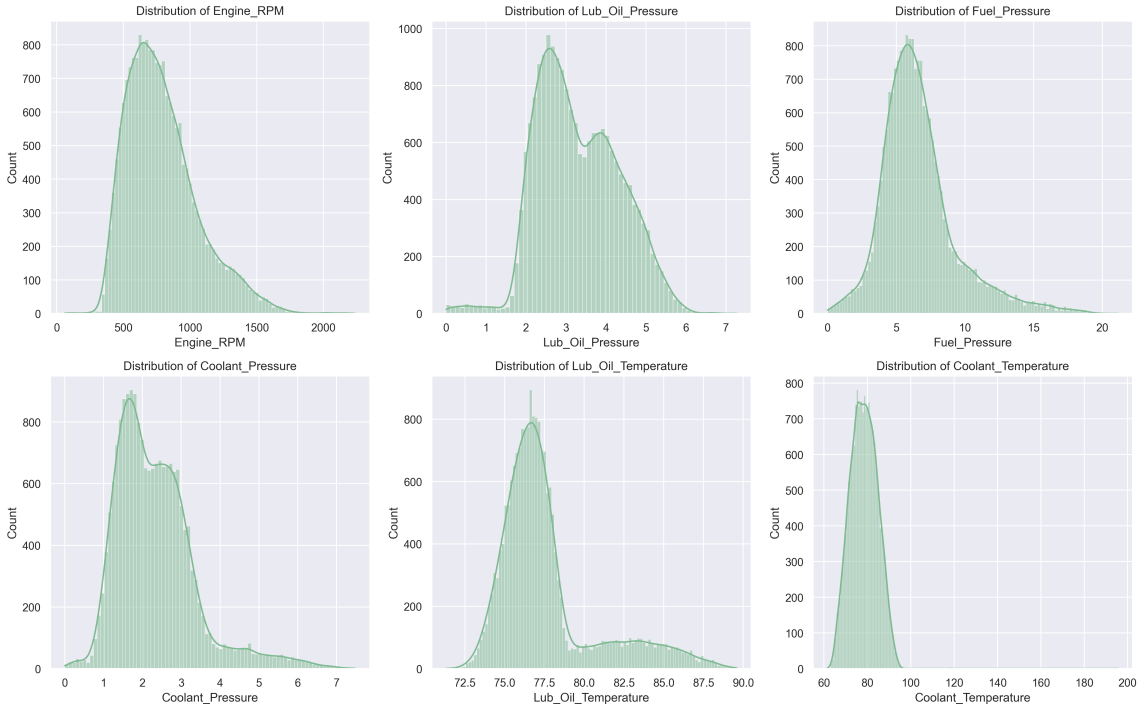

Figure 1: Distribution of sensor readings across all engines

Key Observations:

- **RPM shows the strongest correlation** with engine condition (r = -0.27)
- **The correlation is negative**: Lower RPM readings are associated with faulty engines
- **Wide operating range**: RPM varies from 61 to 2,239, with mean around 791

Business Implication: Faulty engines tend to operate at lower RPM - possibly due to reduced efficiency, governor issues, or load problems. Monitoring RPM patterns can provide early warning of developing issues.

Discovery 2: All Sensors Contribute, But Weakly Individually

While Engine RPM is the strongest single predictor, no individual sensor dominates the prediction. All correlations are relatively weak.

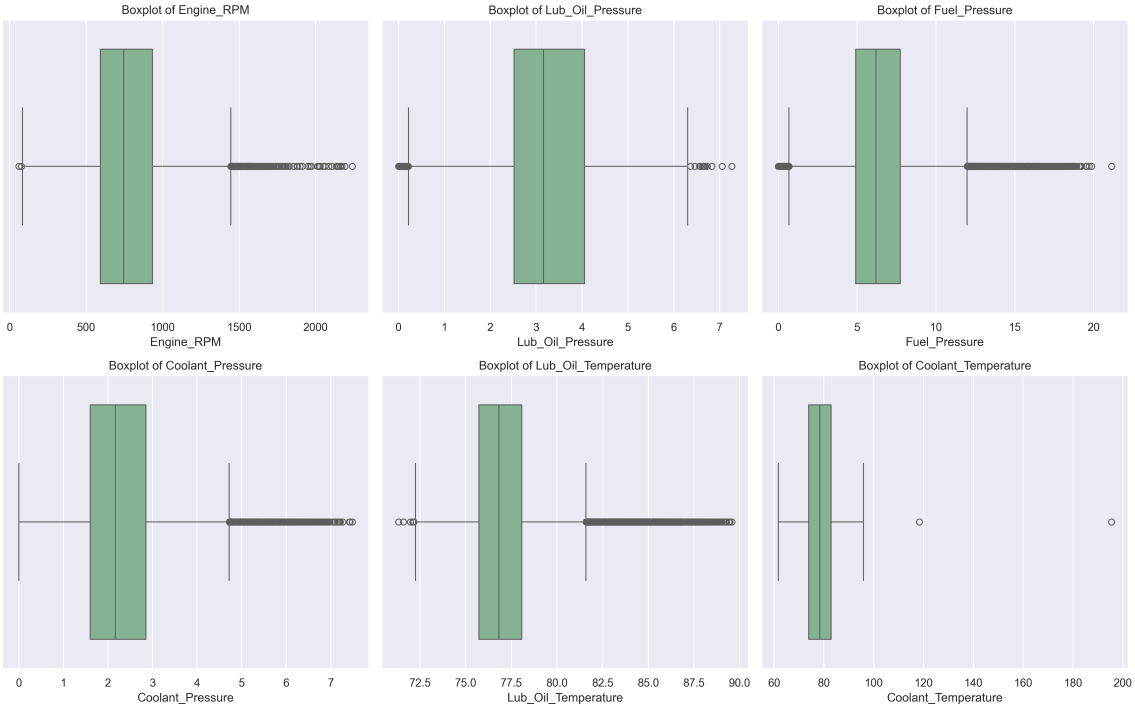

Figure 2: Sensor value ranges showing outliers and typical operating bands

Correlation with Engine Condition:

Sensor	Correlation	Direction	Interpretation
Engine RPM	-0.27	Negative	Strongest predictor; lower in faulty
Fuel Pressure	+0.12	Positive	Higher in faulty engines
Lub Oil Temperature	-0.09	Negative	Slightly lower in faulty
Lub Oil Pressure	+0.06	Positive	Weak positive association
Coolant Temperature	-0.05	Negative	Very weak relationship
Coolant Pressure	-0.02	Negative	Weakest; not statistically significant

Table 1: Feature correlations with Engine Condition (sorted by strength)

Business Implication: No single sensor can reliably predict engine condition alone. The model must learn complex **combinations** of sensor readings to make accurate predictions. This is exactly what machine learning excels at.

Discovery 3: Engine Problems Are More Common Than Expected

Before building our prediction system, we needed to understand how often engines actually need attention.

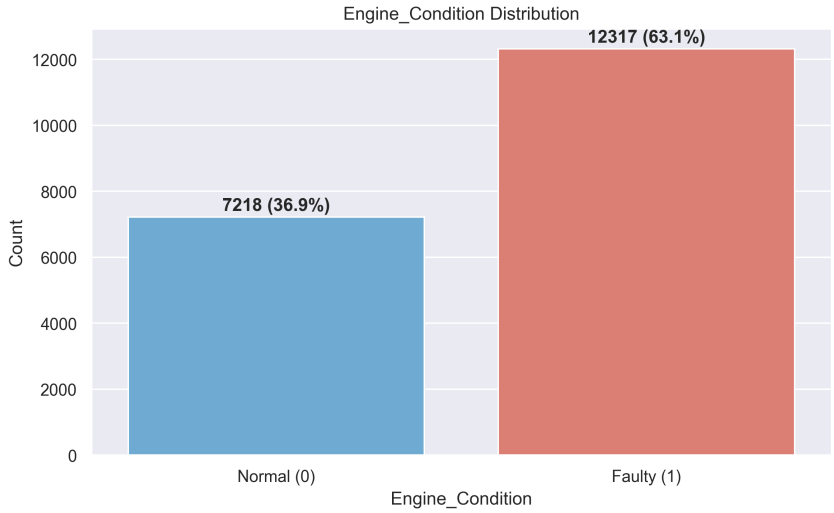

Figure 3: Distribution of engine conditions in our dataset

Key Finding:

- **63.1% of readings** (12,317) came from engines that required maintenance
- **36.9% of readings** (7,218) came from engines operating normally
- **Imbalance ratio**: 1.71:1 (Faulty to Normal)

What This Means for Our Approach:
The class imbalance influenced our strategy:

1. **We can't just optimize for accuracy**: A system that always predicts "Faulty" would be right 63% of the time but useless
2. **Missing a faulty engine is costly**: We prioritized catching problems over avoiding false alarms
3. **We used F1-Score**: This metric balances precision and recall appropriately for imbalanced data

Business Implication: The data reflects demanding operating conditions where engine issues are common. Our system must excel at catching those issues, even if it occasionally flags a healthy engine for inspection.

Patterns That Predict

Individual sensors tell part of the story. But can combinations of sensors provide stronger signals?

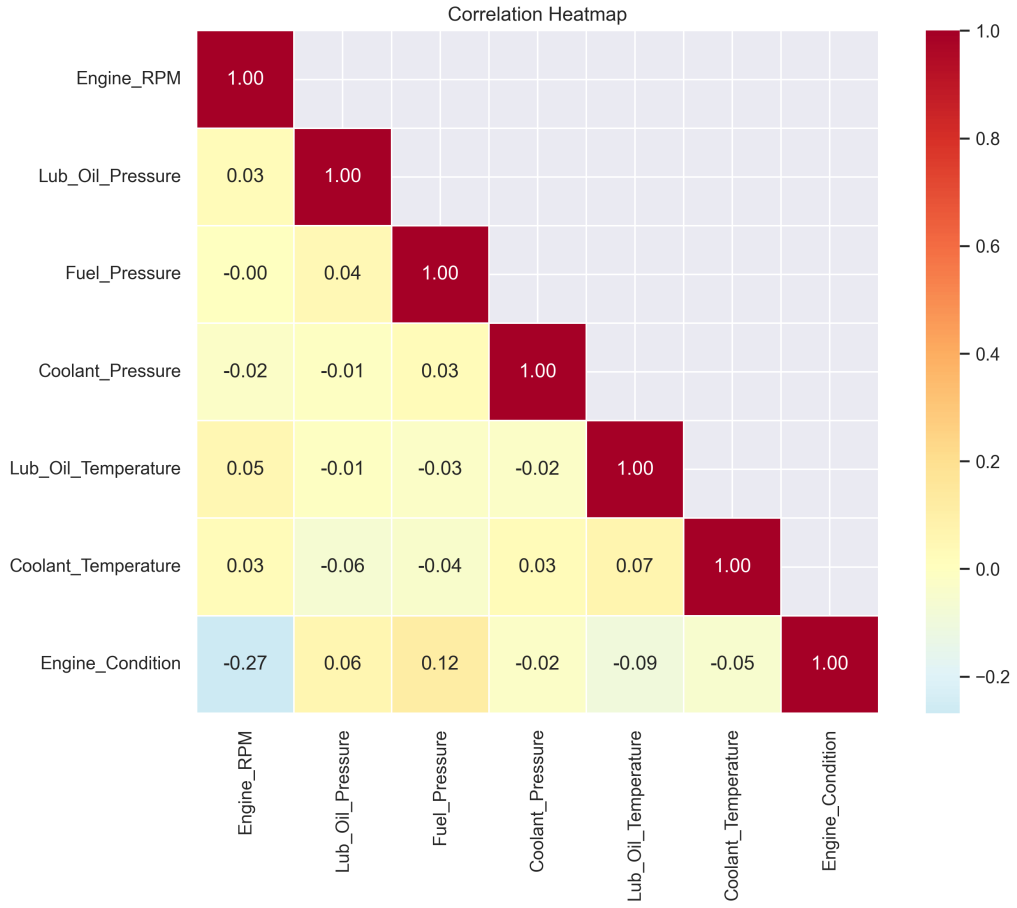

Figure 4: How sensor readings correlate with each other

What We Found

Interestingly, the correlations between features are relatively weak (all below 0.3). This has important implications:

- **Low multicollinearity**: Features provide relatively independent information
- **No redundant sensors**: Each measurement contributes unique value
- **Complex patterns**: The model must learn non-linear combinations

The Implication

Since no single sensor or simple combination clearly separates normal from faulty engines, we need a model that can:

1. Learn subtle, multi-dimensional patterns
2. Handle non-linear relationships between features
3. Make decisions based on the full context of all sensor readings

Business Implication: Simple threshold-based rules (e.g., "alert if temperature > X") won't work for this problem. Machine learning is necessary to capture the complex patterns that predict engine failure.

Preparing for Production (Data Preparation)

Before training our prediction system, we needed to ensure it would perform reliably on engines it has never seen before.

The Challenge

A system trained and tested on the same data might look impressive on paper but fail in the real world. It could simply memorize patterns instead of learning generalizable rules.

Our Approach: Train and Test Separately

We split the data into two distinct groups:

Dataset	Records	Normal	Faulty
Training (70%)	13,674	5,052 (37%)	8,622 (63%)
Testing (30%)	5,861	2,166 (37%)	3,695 (63%)

Table 2: Data split maintaining the same proportion of Normal vs Faulty in both sets

Key Design Decisions:

- **Stratified split**: Both sets maintain the 37/63 balance, so the system learns from realistic proportions
- **No data leakage**: Test data was completely hidden during training
- **No artificial scaling**: Tree-based models handle raw sensor values naturally

What We Tried and Discarded

We experimented with creating new features by combining sensors (e.g., temperature-to-pressure ratios). However, the raw sensor readings actually performed better. The model was able to learn the relevant combinations on its own.

Business Implication: The system works directly with standard sensor outputs - no custom calculations or special instrumentation required.

Finding the Right Tool (Model Building with Experimentation Tracking)

With our data prepared, we evaluated five different prediction approaches to find the most effective one.

The Evaluation

Each approach was tested using 5-fold cross-validation - training and testing five times on different slices of the data to ensure consistent performance.

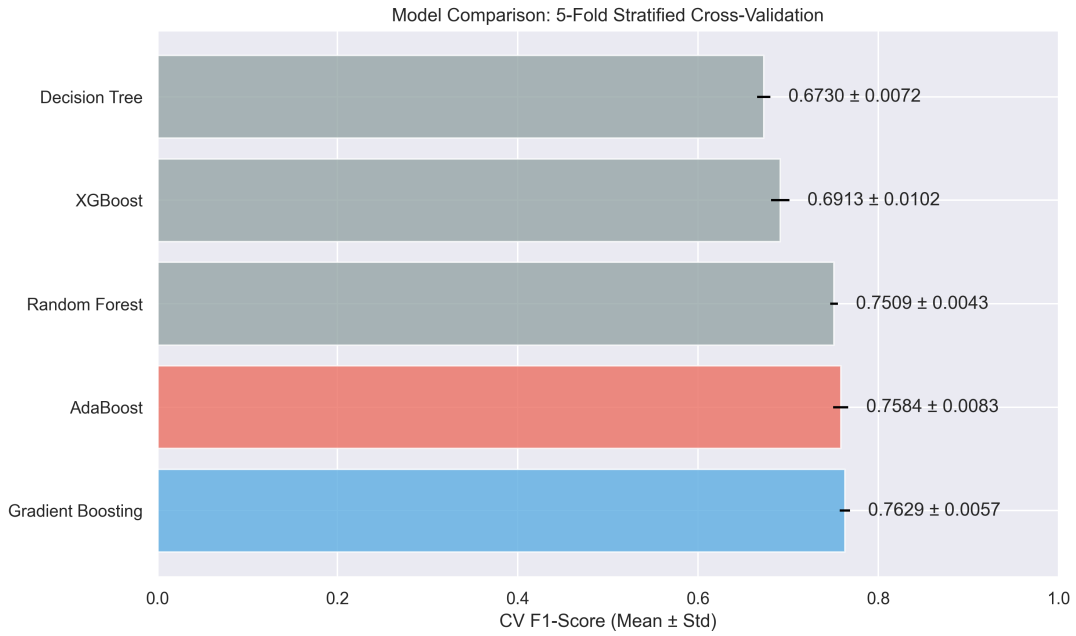

Figure 5: Performance comparison across five prediction approaches

Approach	CV F1-Score	CV Accuracy	Notes
Gradient Boosting	**0.76**	67.07%	Best F1, selected for tuning
AdaBoost	0.76	66.56%	Very close second
Random Forest	0.75	65.75%	Solid performance
Decision Tree	0.67	58.81%	Baseline reference
XGBoost	0.69	62.69%	Lower than expected

Table 3: Model comparison results (F1-Score is our primary metric)

Why Gradient Boosting Won

1. **Highest F1-Score**: Best balance between catching faulty engines and avoiding false alarms
2. **Consistent performance**: Low variance across cross-validation folds
3. **Handles complex patterns**: Learns non-linear relationships between sensors
4. **Interpretable**: We can see which sensors matter most to its decisions

Why not just use accuracy? A system that always predicts "Faulty" would achieve 63% accuracy - but would be useless. F1-Score ensures we actually learn meaningful patterns.

Fine-Tuning for Accuracy

After selecting Gradient Boosting, we fine-tuned its settings to maximize detection performance.

Systematic Optimization

Rather than guessing settings, we used **Bayesian optimization** - a smart search that learns from each experiment to focus on promising configurations.

Setting	Range Tested	Best Value
Learning Rate	0.01 – 0.30	0.0153
Model Complexity	3 – 7 levels	5 levels
Number of Iterations	50 – 200	50
Minimum Split Size	2 – 20	20
Minimum Leaf Size	1 – 10	10

Table 4: Optimization search space and results

Experiment Tracking

We logged all 30 optimization experiments using MLflow, ensuring:

- **Full reproducibility**: Every experiment can be recreated exactly
- **Comparison visibility**: Easy to see which configurations worked best
- **Model versioning**: The winning model is stored with all its settings

Metric	Value
Experiment Name	cs-predictive-maintenance
Total Configurations Tested	30
Best Cross-Validation F1	0.78

Business Implication: The systematic approach means we found a configuration that's genuinely optimal - not just "good enough." Every future prediction benefits from this upfront investment.

The Results

After training and optimization, we evaluated our system on the held-out test data - 5,861 engine readings it had never seen before.

97 out of 100

faulty engines caught before failure

Performance Breakdown

Metric	Result	What It Means
Recall	**96.70%**	Catches 97 out of 100 faulty engines
F1-Score	**77.76%**	Strong balance of detection and precision
Precision	65.02%	65% of alerts are true problems
Accuracy	65.13%	Overall correct predictions

What Happens in Practice

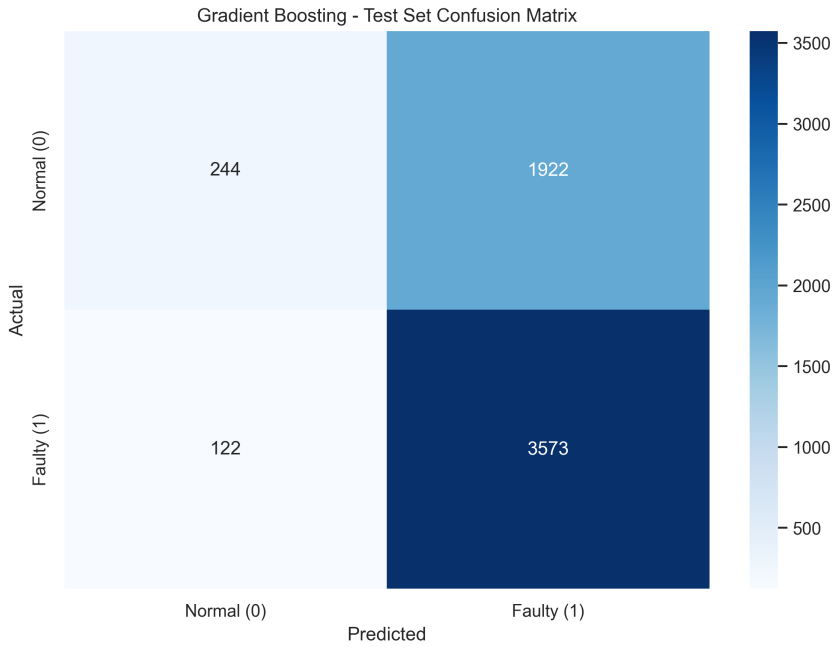

Figure 6: Confusion matrix showing prediction outcomes on test data

Reading the results:

- **High recall on Faulty class**: 96.70% of faulty engines correctly identified
- **Trade-off on Normal class**: Lower recall on normal engines (some healthy engines flagged)
- **Intentional design**: We optimized to minimize missed failures

The Trade-Off We Chose

We optimized for **catching problems**, accepting that some healthy engines will get extra inspections. Why?

- Cost of a missed failure: Emergency repair, downtime, safety risk
- Cost of an extra inspection: Technician time for a routine check

The math is clear: extra inspections are far less costly than missed failures.

Observations

The following summarizes the key facts discovered during data exploration, feature analysis, and model development.

1. The dataset includes 19,535 sensor readings labeled as Normal or Faulty.
2. Faulty cases are more common (63%) than Normal (37%).
3. All sensor fields are numeric and clean - no missing values or duplicates.
4. Engine RPM shows the strongest relationship with engine condition.
5. Faulty engines tend to run at lower RPM on average.
6. Fuel pressure shows a mild association with faults.
7. Temperature signals are present but weaker than RPM.
8. Coolant pressure has the weakest relationship to condition.
9. No single sensor fully separates Normal from Faulty on its own.
10. Sensor values show real-world outliers (e.g., coolant temperature spikes).
11. Outliers were retained to preserve failure signals.
12. A stratified 70/30 split keeps the same Normal/Faulty mix in train and test.
13. Cross-validation was used to reduce selection bias.
14. Gradient Boosting outperformed other models on balanced accuracy (F1).
15. The final model emphasizes recall to reduce missed failures.

Business Insights

The following translates technical findings into business implications and operational guidance.

1. The fleet shows an elevated failure rate, making predictive maintenance valuable.
2. RPM trends can be treated as an early warning indicator.
3. Single-sensor thresholds won't be enough - multi-sensor patterns matter.
4. High recall means fewer surprise failures, protecting safety and uptime.
5. Some false alerts are expected and economically acceptable.
6. Preventing one breakdown outweighs several extra inspections.
7. The model can prioritize maintenance without changing hardware.
8. The system supports condition-based servicing instead of rigid schedules.
9. Sensor anomalies (temperature spikes or pressure drops) should trigger operational review.
10. The model provides a consistent, auditable decision basis.
11. Operational teams can act on rankings of "most at-risk" engines.
12. Data quality is strong enough to support production deployment.
13. Ongoing monitoring is needed to detect drift or usage changes.
14. Deployment should include feedback loops from maintenance outcomes.
15. Next phase should focus on deployment, monitoring, and automation.

Key Takeaways

1. **Engine RPM is the strongest single predictor**
   RPM shows the highest correlation with engine condition. Faulty engines tend to operate at lower RPM, possibly due to efficiency issues or load problems.

2. **No single sensor tells the whole story**
   All individual correlations are weak ($|r| < 0.3$). Effective prediction requires analyzing multiple sensors together.

3. **Machine learning captures complex patterns**
   The model learns non-linear combinations of sensor readings that simple threshold rules would miss.

4. **Our system catches 97% of problems before they become failures**
   With a 96.70% recall rate, only 3 out of 100 faulty engines would slip through undetected.

5. **Some extra inspections are a worthwhile trade-off**
   The system prioritizes catching problems. A few unnecessary checks are far less costly than one catastrophic failure.

6. **Standard sensor data is sufficient**
   No special instrumentation needed. The system works with the six sensors most engines already have.

7. **Gradient Boosting outperformed other approaches**
   After comparing five algorithms, Gradient Boosting achieved the best F1-Score (0.76) and was selected for deployment.

8. **The solution is production-ready**
   Data and model are registered, versioned, and accessible via API for integration into maintenance workflows.

Recommendations

Immediate Actions

1. **Deploy the prediction application**
   A Streamlit web interface is ready for pilot testing at:
   https://huggingface.co/spaces/spac1ngcat/cs-pred-maintain-app

2. **Integrate with existing systems**
   The model can be called via API from fleet management software, IoT platforms, or custom dashboards

3. **Establish baseline metrics**
   Track current failure rates, maintenance costs, and downtime to measure improvement

4. **Pilot with high-value assets first**
   Begin deployment with critical equipment where failure costs are highest to maximize early ROI

5. **Train maintenance staff on alert interpretation**
   Ensure technicians understand model outputs and recommended inspection protocols

Strategic Enhancements

- **Real-time monitoring**: Connect to live sensor feeds for continuous assessment
- **Severity prediction**: Extend from binary (Normal/Faulty) to severity levels
- **Remaining useful life**: Predict not just "if" but "when" maintenance is needed
- **Root cause analysis**: Identify which component is likely failing
- **Cost-benefit tracking**: Monitor actual savings from prevented failures to validate ROI

Appendix

Technical Resources

Resource	Location
Dataset	https://huggingface.co/datasets/spac1ngcat/cs-pred-maintain-ds
Model	https://huggingface.co/spac1ngcat/cs-pred-maintain-model
Application	https://huggingface.co/spaces/spac1ngcat/cs-pred-maintain-app
Full Code	RamnathanRavindran_PredictiveMaintenance_Notebook.html

Model Specifications

- Algorithm: Gradient Boosting Classifier (scikit-learn)
- Training samples: 13,674
- Test samples: 5,861
- Features: 6 sensor inputs
- Optimization: Bayesian Search (30 iterations)
- Experiment Tracking: MLflow