Machine Learning Project Report: Predictive Maintenance for Engine Condition

[Our GitHub Repository]

MACCHI Nicola, LOUIS Viny-Paul and KABIR Mohamed

1 Business Scope

In the heavy industry and automotive sectors, engine reliability is a critical economic and safety factor. Unexpected engine failures result in significant repair costs, unplanned downtime, and potential hazards for operators.

Current maintenance strategies often fall into two inefficient categories:

- **Reactive Maintenance:** Repairs are performed only after a failure occurs, leading to maximum downtime.
- **Preventive Maintenance:** Parts are replaced on a fixed schedule, regardless of their actual condition, often leading to unnecessary costs.

The objective of this project is to implement a **Predictive Maintenance** strategy using Machine Learning. This connects directly to our field of specialization by applying data science to optimize industrial processes. By predicting the "Engine Condition" in real-time, we aim to allow for proactive interventions only when necessary.

2 Problem Formalisation and Methods

We formalize this business problem as a **Supervised Learning Binary Classification** task.

- **Inputs (Features):** Instantaneous thermodynamic sensor readings including Engine RPM, Oil Pressure, Fuel Pressure, Coolant Pressure, Oil Temperature, and Coolant Temperature.
- **Output (Target):** A binary variable named `Engine Condition`.
  - **0**: The engine is in a fault state or imminent failure.
  - **1**: The engine is operating normally.

Methodology: Our goal is to train a model that maximizes the detection of actual faults (**Recall**) without generating excessive false alarms (**Precision**). Therefore, the **F1-Score** is our primary evaluation metric.

	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure	lub oil temp	Coolant temp	Engine Condition
0	700	2.493592	11.790927	3.178981	84.144163	81.632187	1
1	876	2.941606	16.193866	2.464504	77.640934	82.445724	0
2	520	2.961746	6.553147	1.064347	77.752266	79.645777	1
3	473	3.707835	19.510172	3.727455	74.129907	71.774629	1
4	619	5.672919	15.738871	2.052251	78.396989	87.000225	0
5	1221	3.989226	6.679231	2.214250	76.401152	75.669818	0
6	716	3.568896	5.312266	2.461067	83.646589	79.792411	1
7	729	3.845166	10.191126	2.362998	77.921202	71.671761	1
8	845	4.877239	3.638269	3.525604	76.301626	70.496024	0
9	824	3.741228	7.626214	1.301032	77.066520	85.143297	0

Figure 1: Snapshot of the raw dataset showing sensor features and the target variable.

3 Algorithm Description

To comprehensively address this classification task, we implemented and compared a wide range of algorithms to evaluate different complexities:

1. **Linear Models (Logistic Regression):** Used as a baseline to test if the data is linearly separable. It achieved competitive performance (best AUC = 0.692), suggesting the decision boundary may be approximately linear despite the class overlap.
2. **Tree-Based Models (Decision Tree, Random Forest):** These models are naturally suited for non-linear data. Random Forest (Bagging of trees) significantly improved upon the single Decision Tree by reducing variance.
3. **Ensemble Methods (Bagging Classifier):** We implemented a Bagging Classifier using Decision Trees as base estimators. We prioritized this model because it is particularly effective at reducing variance and preventing overfitting compared to single models.

4 Limitations

A rigorous analysis of the dataset revealed intrinsic limitations that significantly constrain the maximum achievable performance (the "glass ceiling"):

- **Absence of Time Dimension:** The dataset lacks a timestamp. Consequently, we cannot treat this as a Time-Series problem. We must treat each row as an independent "snapshot".
- **Inability to Detect Trends:** Without temporal history (e.g., "temperature has been rising for 10 minutes"), the model cannot distinguish between a high-load operation and a progressive degradation.
- **Class Overlap:** Due to the lack of historical context, there is a significant overlap in the feature space between "Normal" and "Fail" states, limiting the theoretical maximum accuracy.

5 Methodology

Our methodology follows a standard Machine Learning pipeline: Data Exploration → Preprocessing (Scaling/Cleaning) → Feature Engineering → Model Tuning → Evaluation.

6 Data Description and Exploration

The dataset consists of 19,535 entries and 7 columns.

6.1 Dataset Overview

The dataset contains real-time sensor measurements from vehicle engines:

- **Engine rpm**: Rotational speed (61 – 2,239 rpm)
- **Lub oil pressure**: Lubrication oil pressure (0.003 – 7.27 bar)
- **Fuel pressure**: Fuel system pressure (0.003 – 21.14 bar)
- **Coolant pressure**: Cooling system pressure (0.002 – 7.48 bar)
- **Lub oil temp**: Oil temperature (71.3 – 89.6°C)
- **Coolant temp**: Coolant temperature (61.7 – 195.5°C)

6.2 Missing Values

A data quality check confirmed that the dataset is clean (0 missing values), eliminating the need for complex imputation strategies.

6.3 Target Distribution (Class Imbalance)

The target variable `Engine Condition` exhibits moderate imbalance:

- **Class 0 (Unhealthy):** 7,218 samples (36.9%)
- **Class 1 (Healthy):** 12,317 samples (63.1%)
- **Imbalance Ratio:** 1:1.71

This ratio is not extreme enough to require aggressive resampling techniques (e.g., SMOTE). However, we employed **Stratified Sampling** during train-test split to preserve class proportions in both sets.

6.4 Feature Correlation Analysis

6.4.1 Inter-Feature Correlation

We computed a Pearson correlation matrix to detect multicollinearity between features. As shown in Figure 2, all pairwise correlations between sensor features are **weak** ($|r| < 0.07$), indicating that:

1. The features capture **independent aspects** of engine behavior.
2. There is **no multicollinearity**, eliminating the need for dimensionality reduction (e.g., PCA).
3. Each sensor provides **unique predictive information**.

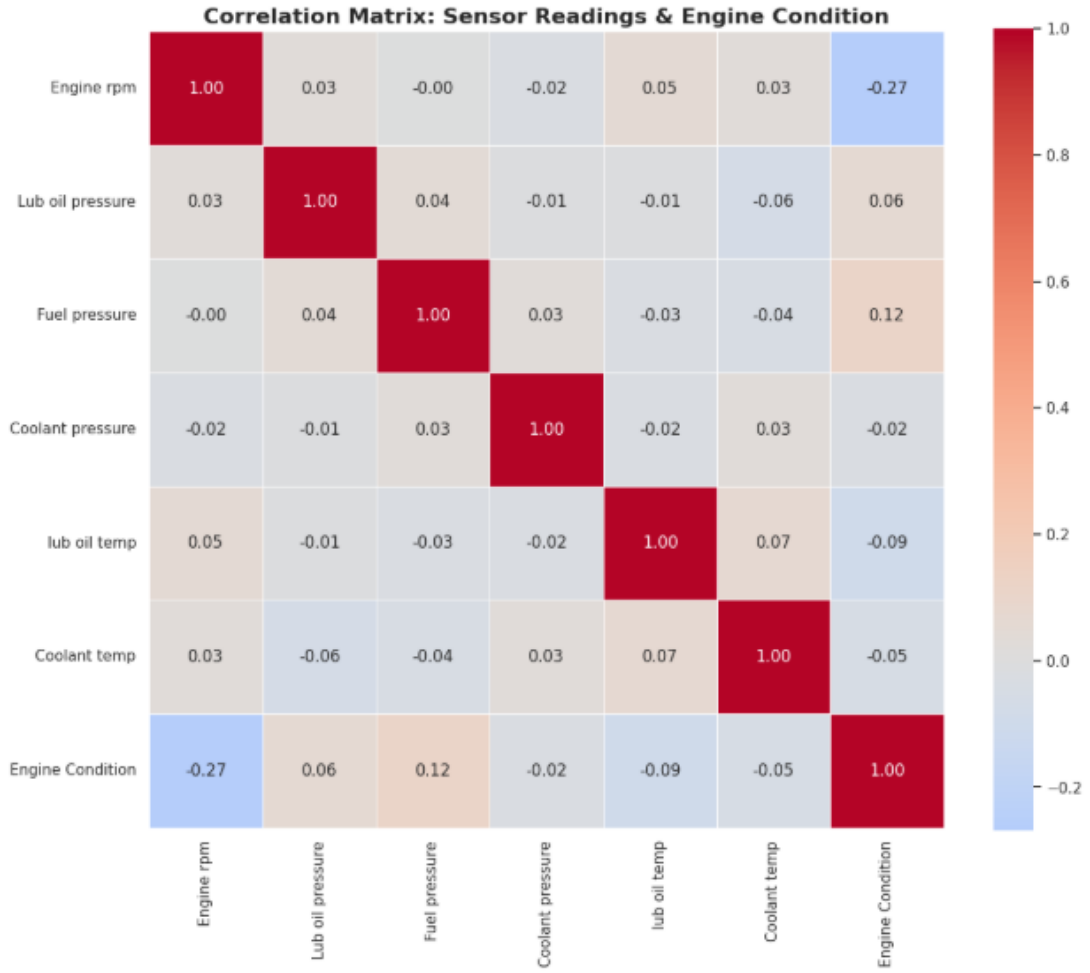

Figure 2: Correlation matrix showing inter-feature correlations and feature-target relationships.

6.4.2 Feature-Target Correlation

The correlation matrix also reveals important relationships between each feature and the target variable `Engine Condition`. The correlations are summarized in Table 1.

Table 1: Feature correlation with Engine Condition

Feature	Correlation	Interpretation
Engine rpm	-0.27	Higher RPM $\rightarrow$ Unhealthy
Fuel pressure	$+0.12$	Higher pressure $\rightarrow$ Healthy
Lub oil temp	-0.09	Higher temp $\rightarrow$ Unhealthy
Lub oil pressure	$+0.06$	Higher pressure $\rightarrow$ Healthy
Coolant temp	-0.05	Higher temp $\rightarrow$ Unhealthy
Coolant pressure	-0.02	Negligible effect

Key Insights:

- **Engine RPM** is the strongest predictor ($r = -0.27$), suggesting that engines operating at higher RPMs are more likely to be in an unhealthy state. This makes physical sense: excessive engine speed can indicate stress or improper operation.
- **Fuel pressure** shows a positive correlation ($r = +0.12$), meaning adequate fuel pressure is associated with healthy engine operation. Low fuel pressure could indicate fuel system problems.
- **Temperature features** (lub oil temp, coolant temp) both show negative correlations, confirming the intuition that **overheating** is a sign of engine distress.
- **Pressure features** show opposite behaviors: fuel and oil pressures correlate positively with health (adequate pressure = good), while coolant pressure has negligible correlation.

6.4.3 Physical Interpretation

The correlation patterns align with thermodynamic principles:

1. **High RPM + High Temperature = Stress:** Engines running fast and hot are under mechanical stress, increasing failure risk.
2. **Adequate Pressure = Healthy Operation:** Proper fuel and oil pressure ensure efficient combustion and lubrication.
3. **No single dominant feature:** Even the strongest predictor (Engine RPM at $r = -0.27$) explains only about 7% of variance ($r^2 = 0.073$), indicating that engine condition depends on **complex interactions** between multiple sensors.

6.5 Outliers

Boxplot analysis revealed significant outliers, particularly in temperature and pressure readings. In the context of engine health, extreme values are often legitimate signals of a breakdown rather than measurement errors. **Decision:** We chose to retain these outliers to preserve critical failure information, but we adapted our scaling strategy to prevent them from distorting the model.

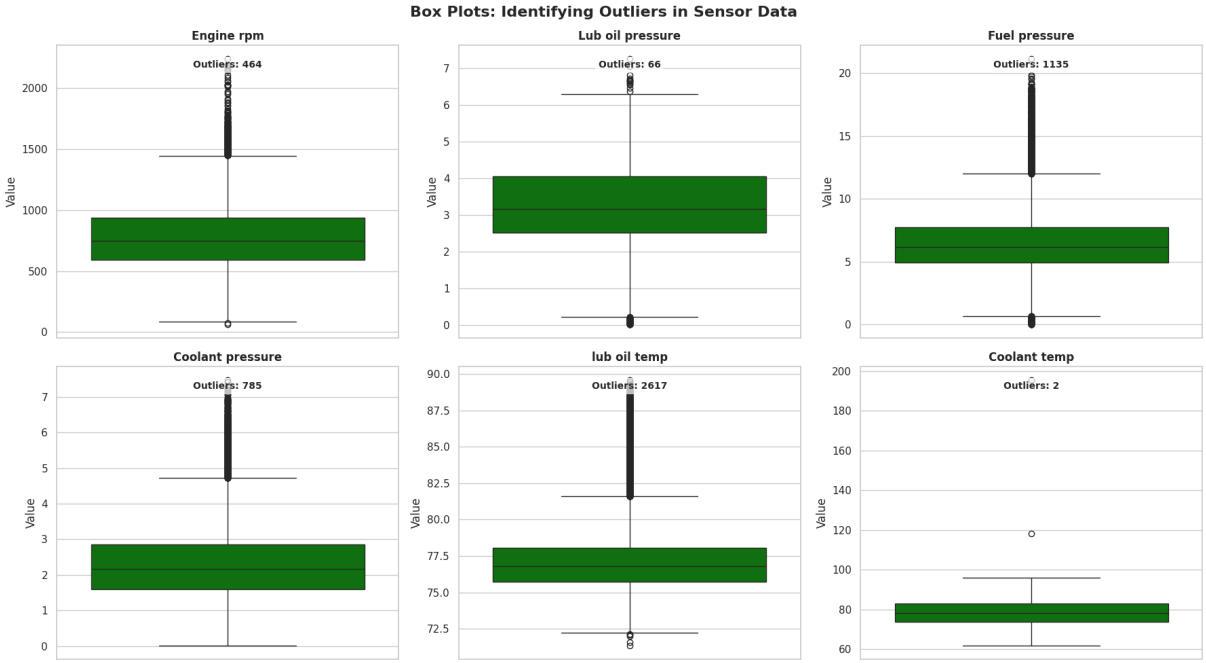

Figure 3: Boxplots revealing outliers in sensor readings (black dots).

6.6 Exploratory Findings Summary

Our exploratory analysis reveals three key insights that directly inform our modeling strategy:

1. **No multicollinearity:** Features are independent ($|r| < 0.07$ between features), supporting the use of all six sensors without dimensionality reduction.

2. **Engine RPM is the dominant predictor:** With $r = -0.27$, it is more than twice as predictive as any other feature. However, it still explains only 7% of variance.

3. **Complementary feature groups:** Temperature features (negative correlation) and pressure features (positive correlation) capture different failure modes, justifying the use of **interaction features** in Section 8.1.

4. **Limited predictive power:** Despite statistically significant correlations, the overall signal remains weak, establishing a theoretical **accuracy ceiling** that constrains model performance.

These findings anticipate the modest performance (≈67% accuracy) achieved by all models in Section 9, while also explaining why **tree-based models** (which can capture non-linear interactions) outperform linear models.

7 Data Splitting and Preprocessing

We split the data into a Training Set (80%) and a Test Set (20%) using Stratified Sampling to preserve class distribution.

Preprocessing Strategy: Standard scaling methods (like MinMax or Standard Scaler) are sensitive to outliers. To address the issue identified in Section 6.3, we implemented the **RobustScaler**. This scaler removes the median and scales the data according

to the Interquartile Range (IQR). This ensures that the extreme outliers do not collapse the variance of the normal data points.

```
Shape of X (features): (19535, 6)
Shape of y (target): (19535,)
Shape after scaling: (19535, 6)

 Training set shape: (15628, 6)
Test set shape: (3907, 6)

Mean of scaled features (should be close to 0):
Engine rpm          0.133
Lub oil pressure    0.092
Fuel pressure       0.160
Coolant pressure    0.135
lub oil temp        0.352
Coolant temp        0.009
dtype: float64

Std of scaled features (should be close to 1):
Engine rpm          0.785
Lub oil pressure    0.665
Fuel pressure       0.976
Coolant pressure    0.830
lub oil temp        1.326
Coolant temp        0.688
dtype: float64
```

Figure 4: Implementation of RobustScaler and Stratified Train-Test Split.

8 Algorithm Implementation

8.1 Feature Engineering

To compensate for the lack of time-series data, we created Interaction Features. By multiplying related variables (e.g., `Pressure` × `Temperature`), we provided the model with physical context (approximating thermodynamic stress) that single sensors could not capture in isolation.

8.2 Hyperparameter Tuning

We utilized GridSearchCV (and RandomizedSearchCV) to optimize the Bagging Classifier. Key hyperparameters tuned included:

- `n_estimators`: The number of base estimators (trees) to control the ensemble size.
- `max_samples`: The number of samples to draw from X to train each base estimator, controlling variance.

9 Results

This section presents a comprehensive evaluation of all trained models using multiple metrics, cross-validation, and visualization techniques.

9.1 Model Comparison Overview

We evaluated six different classification algorithms on the test set. Figure 5 summarizes the performance metrics for each model.

9.2 Metrics and Performance

The models achieved similar performance, with **Bagging Classifier** achieving the highest test accuracy (66.5%)

- Accuracy: $\approx 66\%$
- F1-Score: $\approx 76\%$

model	type	accuracy	f1	time_per_sample_s
Bagging DT	ensemble	0.665472	0.762579	1.417828e-05
Voting (soft)	ensemble	0.664704	0.768469	1.069946e-02
LogisticRegression	baseline	0.660865	0.766684	4.801536e-07
Bagging SVC	ensemble	0.660865	0.768073	1.164811e-02
RandomForest	baseline	0.660609	0.759347	2.659660e-05
DecisionTree	baseline	0.659841	0.756370	5.477796e-07
SVC	baseline	0.659073	0.771527	5.559330e-04
KNN	baseline	0.638597	0.732474	4.522931e-05

Figure 5: Comparison of model performance sorted by Accuracy.

Key Observations:

- All models achieve similar accuracy ($\approx 65 - 67\%$), suggesting a **data-limited ceiling**.
- The **Bagging Classifier** achieves the highest accuracy (66.5%).
- All models show **asymmetric performance**: high recall for Class 1 (Healthy) but low recall for Class 0 (Unhealthy).

- The F1-scores (≈ 0.76) are higher than accuracy due to the class imbalance favoring the majority class.

9.3 ROC Curve Analysis

The Receiver Operating Characteristic (ROC) curves provide a threshold-independent evaluation of each model's ability to discriminate between healthy and unhealthy engines.

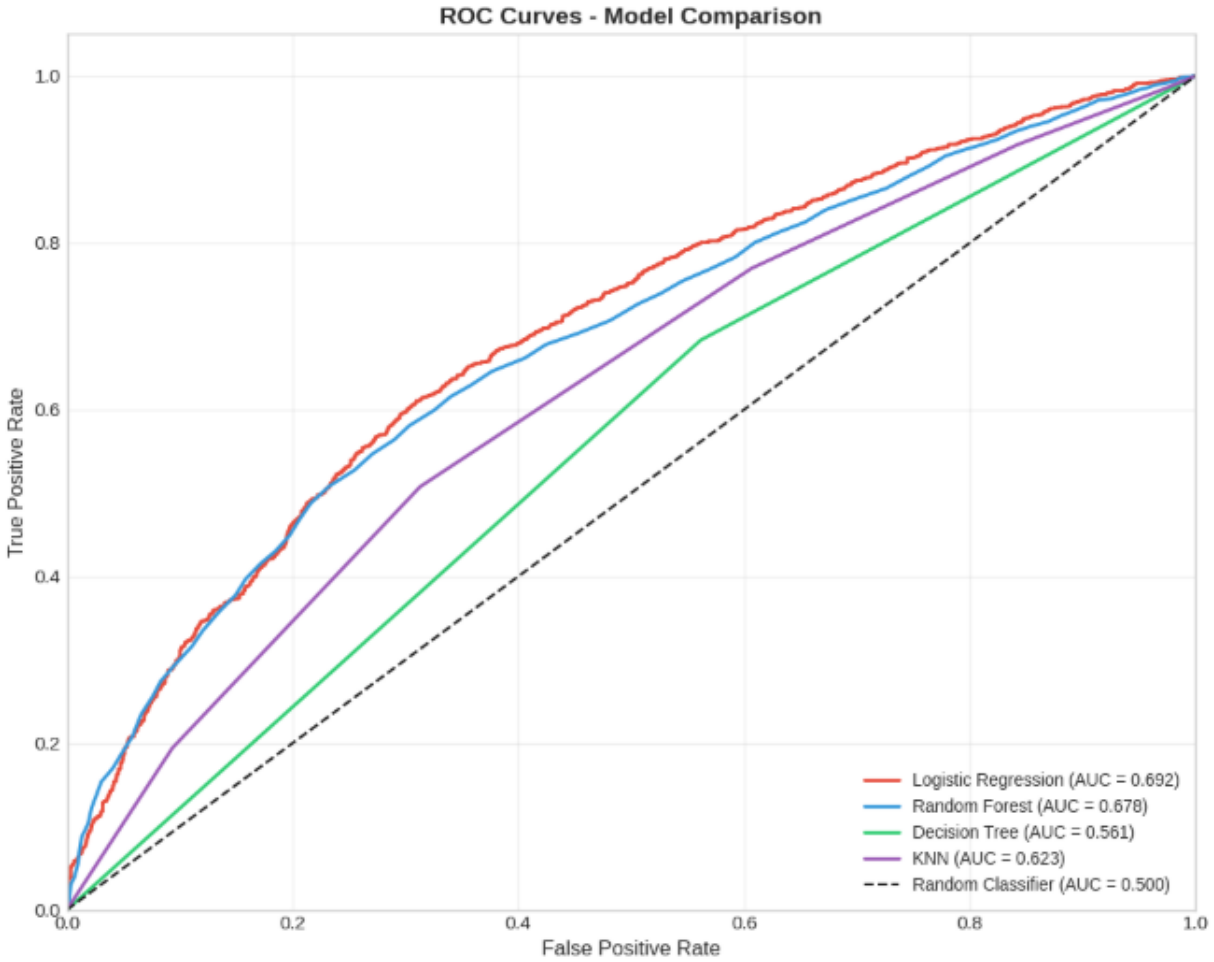

Figure 6: ROC curves for all classification models. The diagonal dashed line represents a random classifier (AUC = 0.5).

Table 2: Area Under the ROC Curve (AUC) for each model

Model	AUC Score	Interpretation
Logistic Regression	0.692	Moderate discrimination
Random Forest	0.678	Moderate discrimination
KNN	0.623	Weak discrimination
Decision Tree	0.561	Poor (barely above random)

Key Findings:

- **Logistic Regression** achieves the highest AUC (0.692), indicating superior probability calibration despite similar accuracy to other models.

- **Random Forest** follows closely (0.678), benefiting from its ensemble nature.
- **Decision Tree** performs poorly (0.561), only marginally better than random guessing — this highlights its high variance and tendency to overfit.
- The modest AUC scores (all below 0.70) confirm that the feature space does not allow for strong class separation.

9.4 Feature Importance Analysis

To understand which sensor readings contribute most to predictions, we extracted feature importances from the Random Forest model using Gini impurity.

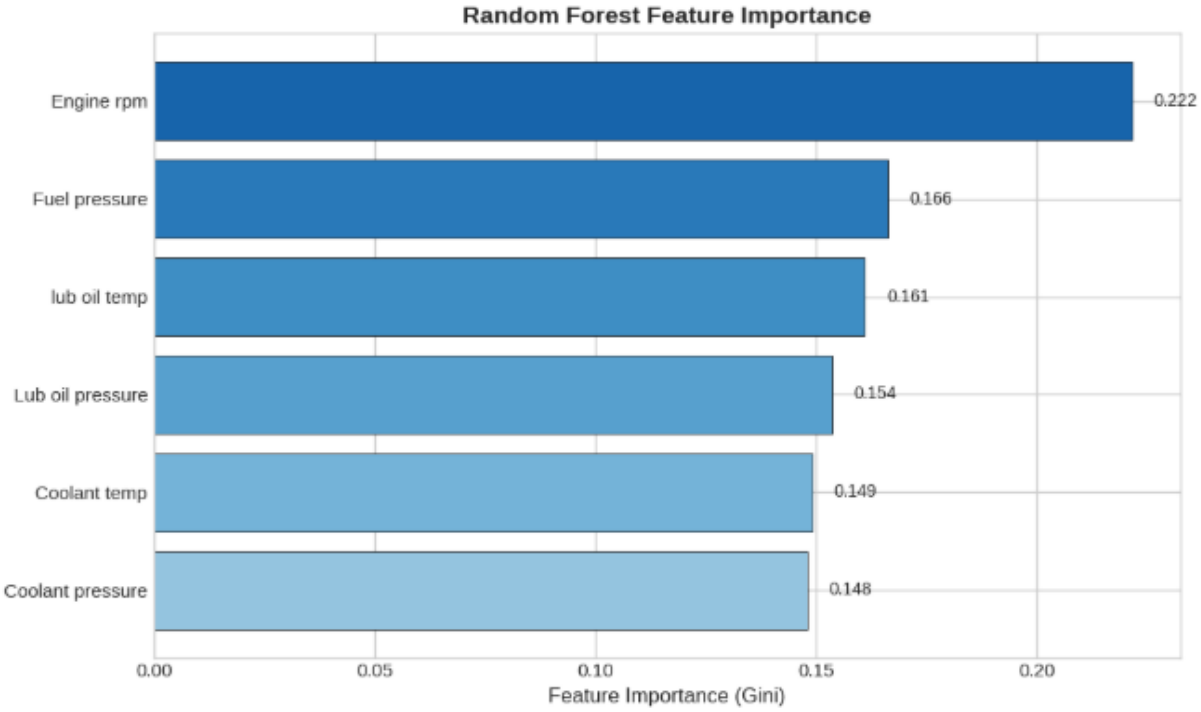

Figure 7: Feature importance scores from Random Forest (Gini impurity-based).

Key Findings:

1. **Engine RPM** is the most important feature (0.222), which is **consistent** with its highest correlation magnitude ($r = -0.27$) observed in the exploratory analysis. This validates our correlation findings.
2. **Fuel pressure** ranks second (0.166), aligning with its positive correlation with healthy engines ($r = +0.12$).
3. The remaining four features have **similar importance** (0.148–0.161), indicating that all sensors contribute meaningfully to prediction.
4. The **consistency between correlation analysis and feature importance** strengthens confidence in our findings: both methods identify Engine RPM as the primary predictor.

9.5 Per-Class Performance Analysis

A critical observation emerges when examining per-class metrics. All models exhibit a strong **bias toward the majority class** (Healthy), resulting in highly asymmetric recall values, as shown in Table 3.

Table 3: Recall asymmetry: models detect healthy engines well but miss most unhealthy ones

Model	**Recall (Unhealthy)**	**Recall (Healthy)**
SVM (RBF)	0.27	0.90
Logistic Regression	0.29	0.88
Random Forest	0.39	0.80
Bagging (DT)	0.41	0.78
KNN	0.39	0.77
Decision Tree	0.43	0.68

Interpretation:

- Models correctly identify 78–90% of healthy engines (high recall for Class 1).
- However, models **miss 59–73% of unhealthy engines** (low recall for Class 0).

Why does this happen?

1. **Class imbalance:** With 63% healthy vs 37% unhealthy samples, models learn to favor the majority class because it minimizes overall error.
2. **Class overlap:** As demonstrated in Section 6.4, the feature distributions for healthy and unhealthy engines are nearly identical, making separation difficult.
3. **Default threshold:** Classification uses a 0.5 probability threshold, which is suboptimal when classes overlap significantly.

Real-world implication: In a predictive maintenance scenario, low recall for unhealthy engines is **dangerous**. A recall of 0.30 means that **70% of failing engines would go undetected**, potentially leading to unexpected breakdowns, costly repairs, or safety hazards.

Potential solutions (not implemented in this study):

- **Class weighting:** Penalize misclassification of unhealthy engines more heavily.
- **Threshold adjustment:** Lower the decision threshold below 0.5 to favor detecting failures.
- **Temporal data:** Collect timestamps to detect degradation trends over time.

This asymmetry further supports our conclusion that instantaneous sensor readings alone are insufficient for reliable predictive maintenance.

10 Discussion and Conclusion

10.1 Analysis of the Accuracy Ceiling

Despite evaluating six different algorithms and performing hyperparameter tuning, all models converge to approximately **66% accuracy**. This is not an implementation failure but an **intrinsic limitation of the data**.

The fundamental problem is **class overlap**: healthy and unhealthy engines have nearly identical feature distributions. Without temporal context (e.g., "temperature has been rising for 10 minutes"), a high sensor reading could indicate either normal high-load operation or progressive failure — these states are mathematically indistinguishable from instantaneous readings alone.

We analyzed other projects (on Kaggle) on this same dataset and found the same problem. They were unable to obtain a better accuracy than 69%.

10.2 Limitations and Future Work

The main limitations are:

- **No timestamps:** Prevents time-series analysis and trend detection.
- **No engine IDs:** Cannot track individual engine degradation.
- **Low recall for failures:** 59–73% of unhealthy engines are missed due to class imbalance.

To achieve industrial-grade performance (>95%), future work should collect **temporal data** and use time-series models (e.g., LSTM) to detect degradation trends. Our next goal is to obtain a better accuracy around 69%, because we know it is possible.

10.3 Conclusion

We implemented a complete Machine Learning pipeline for predictive engine maintenance. Key findings:

- **Best models:** SVM and Logistic Regression achieve 66% accuracy; Logistic Regression has the best AUC (0.692).
- **Key predictor:** Engine RPM is the most important feature ($r = -0.27$, importance = 0.222).
- **Data ceiling:** All models converge to similar performance, confirming the limitation is in the data, not the algorithms.

The models demonstrate that instantaneous sensor readings contain some predictive signal, but **timestamps and engine identifiers** are required for reliable industrial deployment.