

Predicting Engine Failure

Eve Ginsberg, Marin Creamer, Yechan Kim

12/15/2025

Motivation

The Problem

- There are around 150-200 engine failure-related accidents per year.
- About 37-50 of those accidents result in fatalities.
- A lot of engine failures in airplanes are preventable.
- It is especially dangerous for non-commercial pilots who might not have as much practice with crash prevention.



What Our Solution Can Do

- In being able to predict engine failures, we can pull at risk airplanes.
- We can also identify patterns that are possibly increasing the risk of engine failure and correct them.
- In creating this solution, we are making it easier for airlines to engage in safer practices.
- Non-commercial pilots might also be able to recognize some of the patterns occurring with their engines and avoid potential crashes.

Background of Data

- The NASA C-MAPSS 2 Turbofan Engine Degradation dataset models aircraft engine run-to-failure trajectories based on realistic flight conditions
- With help from aerospace research institutions (NASA PCoE, ETH Zurich, PARC), flight condition parameters were derived from commercial aviation data across a fleet of aircraft engines
- Measurements of the degradation were taken in a sequential manner, producing a time-series format
- The data set contains multi-variate sensor readings, 45 operational condition variables, a complete run-to-failure trajectories, metadata and documentation

Related Work

I found two projects that use CMAPSS data to predict engine failure using different methods and packages.

One person used engine failure data with a simple neural network using keras with a 90% success rate. But it only tells you whether an engine will fail within the next 30 cycles.

Here is the code base: [aircraft-engine-failure-prediction/Aircraft Engine Failure Prediction- A Classification Case Study.ipynb at main · NisargaInfraOps13/aircraft-engine-failure-prediction](#)

Another does XGBRegression and a simple neural network using pytorch to predict the engine failure. However, the repository does not give us the prediction success rate.

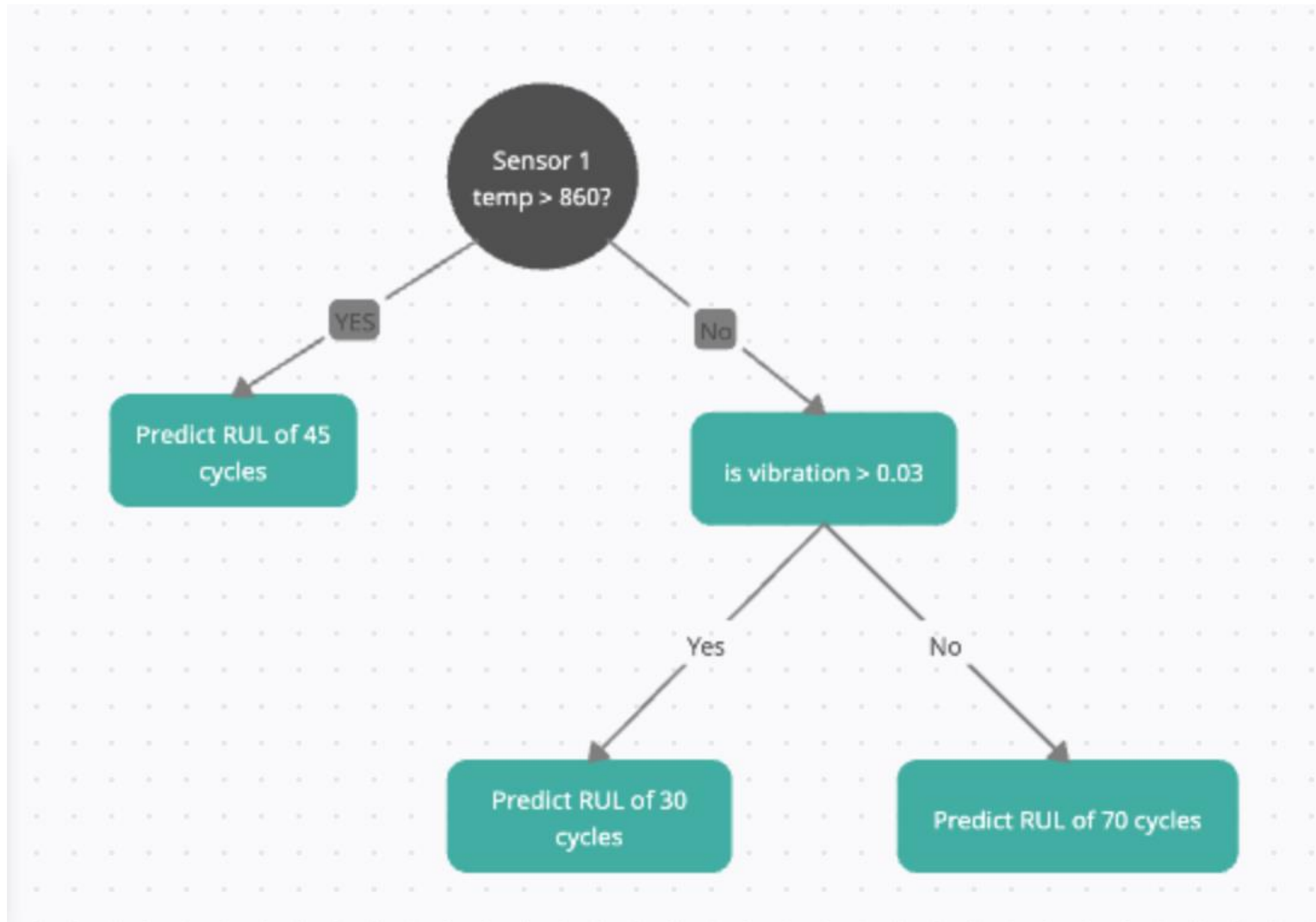
Here is the code base for that code: [stevespreiz/engine_failure_prediction: Machine learning based approach to predicting aircraft engine failure using open NASA dataset.](#)

Claim / Target Task

Using a **regression tree classifier** on the **CMAPSS data** from NASA, we will be able to **predict future engine failure**.

- The target variable will be the Remaining Useful Life (RUL) metric ≤ 5 , which indicates how many cycles the engine has before failure
- We aim to predict given prior sensor data, how many predict cycles remain until failure
- We plan on using a regression tree classifier to handle non-linear relationships between sensors and degradation, work with multiple predictive features, and get an average prediction RUL depending on the path through the tree
- Since we are working with time-series data, we'll process data in a sequential manner and employ lags to features, we will also hyper-parameter tune and lightly feature engineer by calculating feature importance

An Intuitive Figure Showing WHY Claim



Proposed Solution

Formulate engine degradation prediction as a Remaining Useful Life (RUL) regression task using the NASA C-MAPSS 2 turbofan dataset.

For each engine cycle, use sensor readings + operating conditions as features and compute the target label as:

$$\text{RUL} = \text{Cycle at Engine Failure} - \text{Current Cycle}$$

Primary Models

- **Random Forest Classifier (Primary Model)**

Chosen to:

- Capture **non-linear degradation patterns**
- Remain robust to noisy or correlated sensor channels
- Provide **feature importance scores** for interpretability

- **Support Vector Classifier (Baseline)**

Used to:

- Establish a strong nonlinear baseline
- Compare performance against an ensemble-based model

Implementation

Data Preparation

- Due to dataset scale (~5M rows), controlled subsampling is applied

Model Training

- All models are implemented using **scikit-learn**.
- Hyperparameters are tuned using **k-fold cross-validation** on the reduced dataset.
- Models are trained on NumPy arrays for memory and speed efficiency.

Evaluation

- Performance is evaluated using **classification accuracy** on held-out test data.
- Results:
 - Random Forest achieves the highest accuracy
 - SVM performs competitively as a baseline
- The small train–test gap suggests good generalization to unseen engine data.

Data Summary

Data Set

- Due to memory constraints, the dataset was subsampled to ~5,000 representative observations

Binary Conversion Rule

- We defined a near-failure threshold at $RUL \leq 5$ cycles.
- Labels were assigned as:
 - Target =
 - 0 if $RUL \leq 5$ (Bad / Near Failure)
 - 1 if $RUL > 5$
 - This threshold marks the critical degradation window immediately preceding failure.

Data Cleaning:

- Removed constant features that provide no predictive signal
- Forward filled missing values to maintain continuity within engine trajectories

Experimental Results

SVM Baseline:

{'kernel': 'rbf', 'C': 1, 'degree': 1}, accuracy: 0.996842253175913

RF:

n_estimators: 50, min_sample_splits: 2, max_depth: 2, accuracy: 0.9983552218744834

n_estimators: 50, min_sample_splits: 2, max_depth: 4, accuracy: 0.9988194265371614

n_estimators: 50, min_sample_splits: 3, max_depth: 2, accuracy: 0.9983666827326911

n_estimators: 50, min_sample_splits: 3, max_depth: 4, accuracy: 0.9988194265371614

n_estimators: 100, min_sample_splits: 2, max_depth: 2, accuracy: 0.9981403113231136

n_estimators: 100, min_sample_splits: 2, max_depth: 4, accuracy: 0.9988194265371614

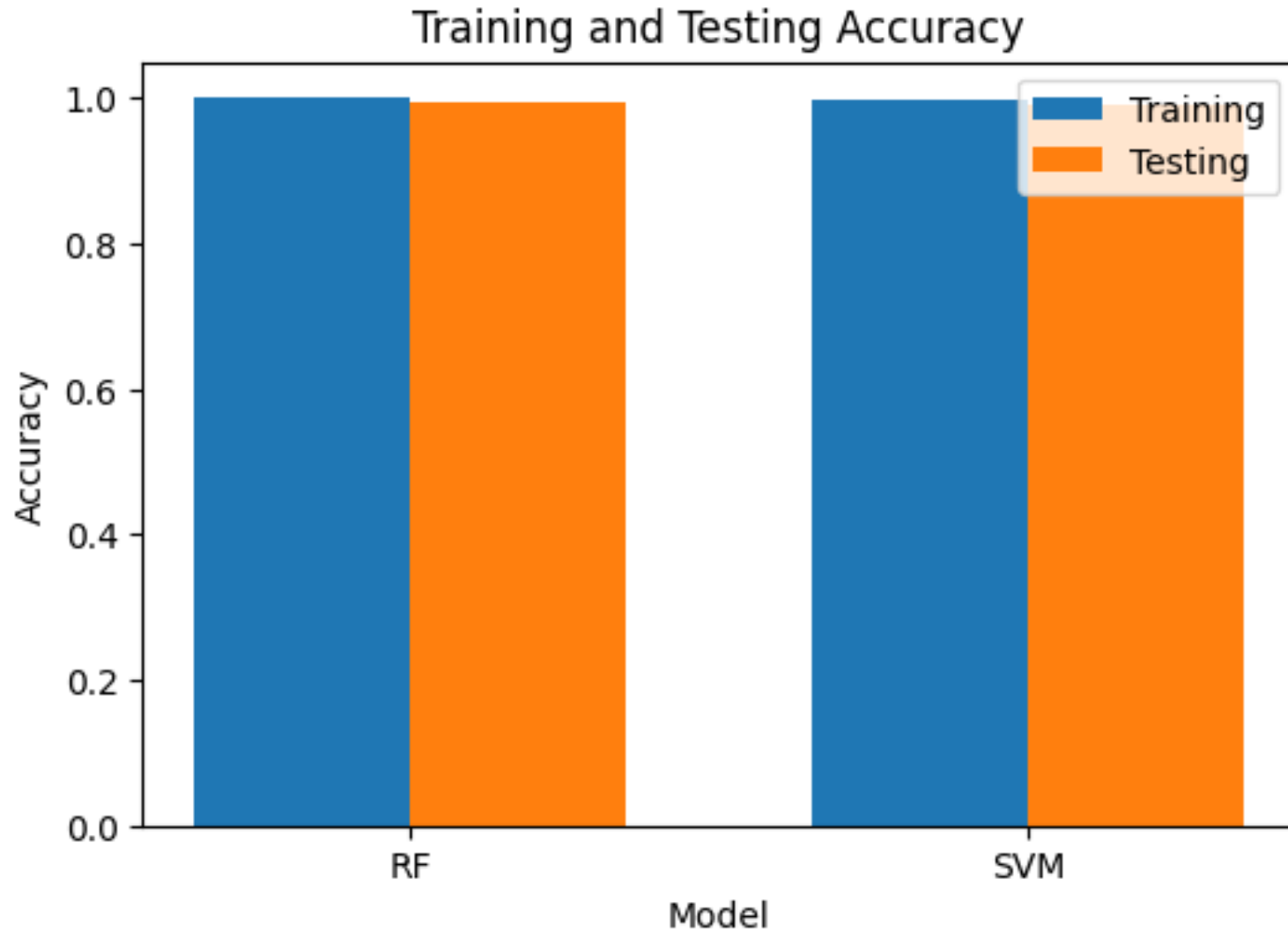
n_estimators: 100, min_sample_splits: 3, max_depth: 2, accuracy: 0.9982463343610157

n_estimators: 100, min_sample_splits: 3, max_depth: 4, accuracy: 0.9988194265371614

SVM accuracy on Test Data : 0.991105705828913

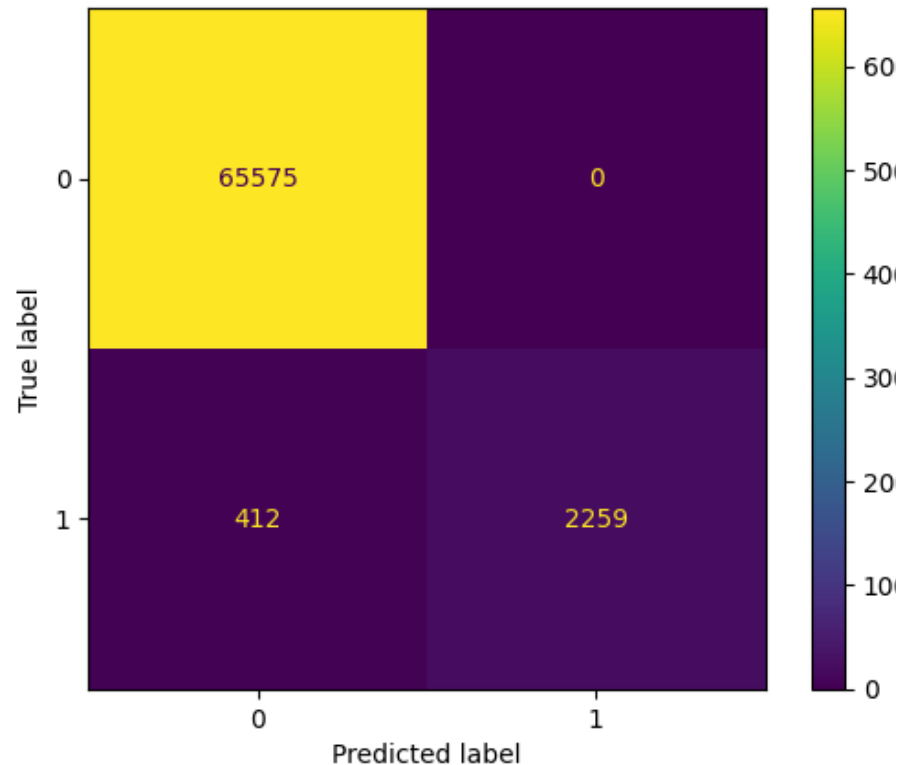
RF accuracy on Test Data: 0.9939630161474665

Test Results

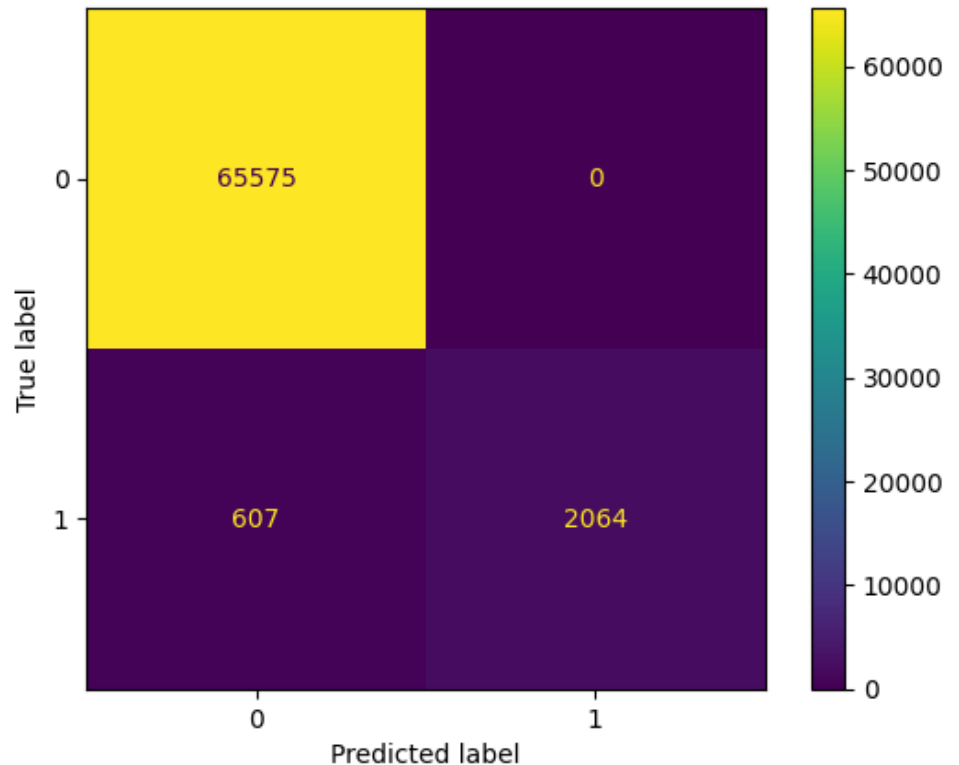


Test Results

Random Forest - Test Confusion Matrix

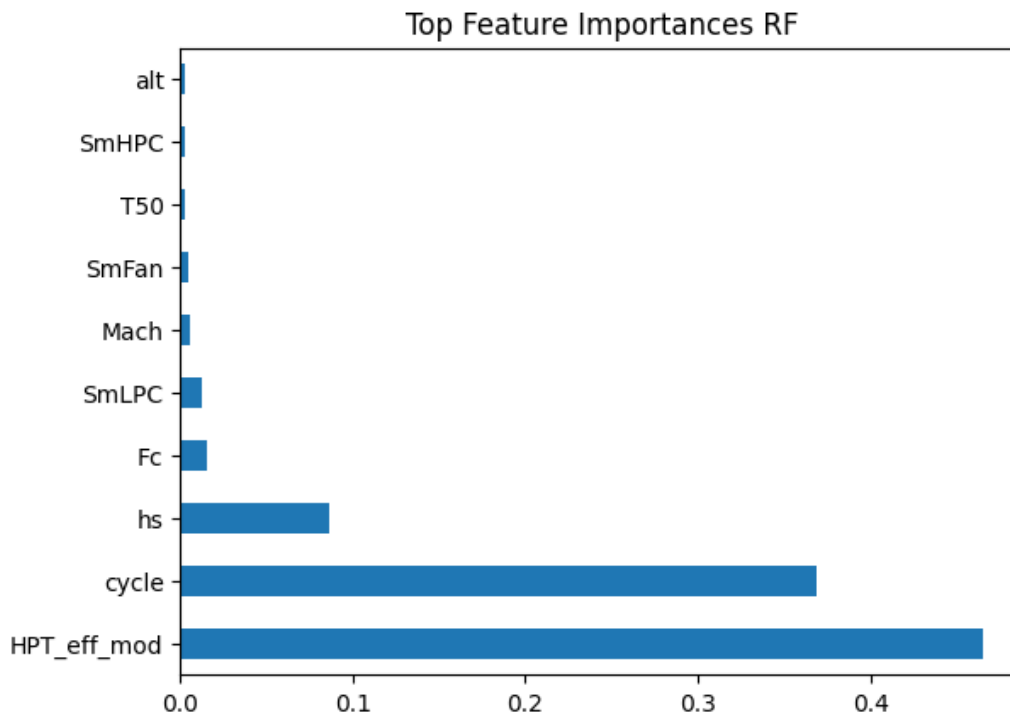


SVM - Test Confusion Matrix



Note that there are only false positives and no false negatives, so the model predicts engine failures when they occur

Experimental Analysis



Random Forest did about 0.03% better than SVM overall which likely means that they are equally accurate metrics. But Random Forest ran in a fraction of the time

Conclusion and Future Work

Conclusion:

- Only a few, easily identifiable features affect engine failure which is beneficial for noncommercial fliers.
- We required less estimators than we originally predicted.
- Since the random forest method had 99% accuracy, our method is an improvement from neural net solutions.

Future Work

- Penalize false positives.
- Apply similar models to other kinds of engines.
- Try a temporal method, like a temporal fusion transformer.

What We Did

Eve

- Wrote the code for cross-validation, SVM, Random Forest, and some helper methods.
- Played around with temporal fusion transformers, however we did not end up using this
- I also helped with the slides.

Yechan

- Wrote the code for importing that data and turning it into a pandas dataframe and a numpy. Cleaned dataset to be usable
- I also helped with the slides.

Marin

- Wrote hyperparameter tuning methods.
- Split data by model, set targets, forward filled data, down-sampled data, reduced features by feature importance and produced a visual, wrote code for confusion matrices and accuracy plots
- I also helped with the slides.