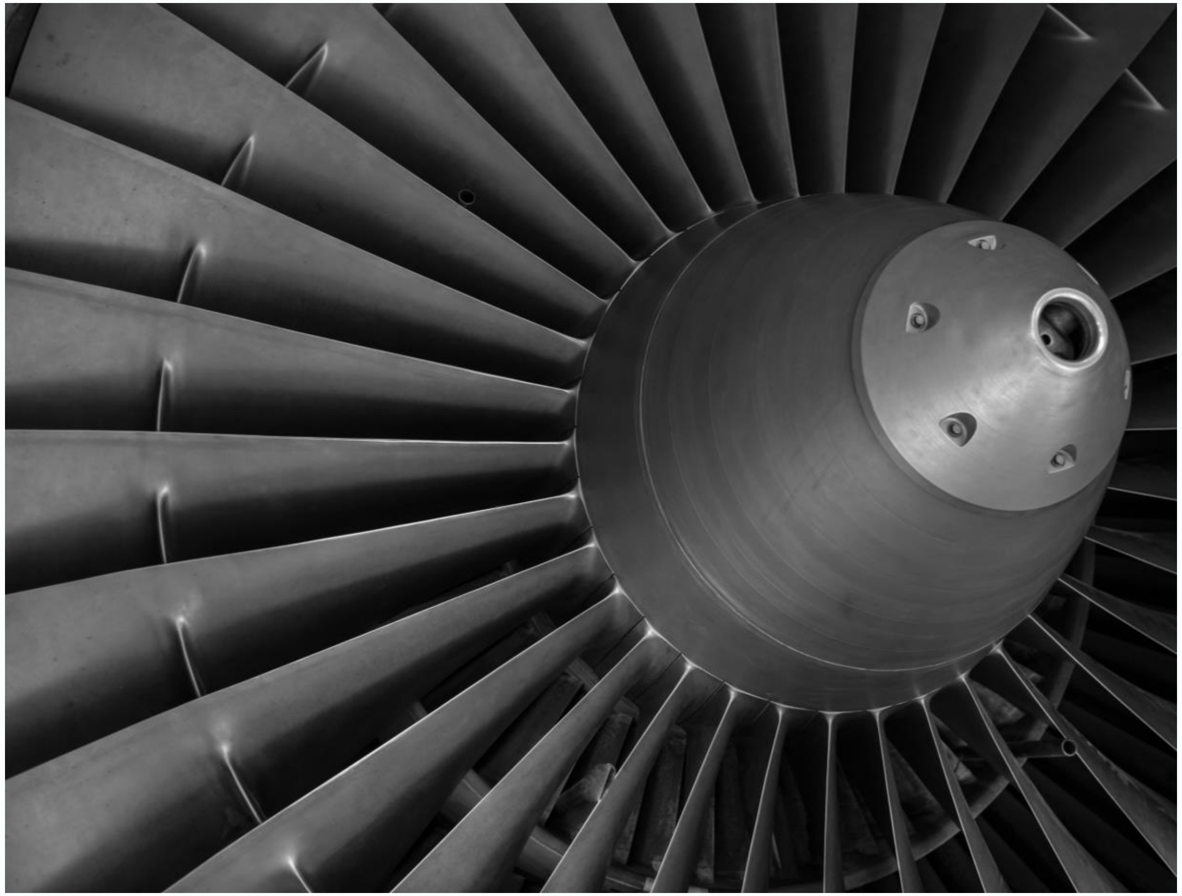
# Service Cost Optimization

## Predictive Maintenance for Turbofan Jet Engine

Andrea Fantini

# Context



Jet Engines undergo maintenance after a certain number of *flight cycles.* There are two kinds of service events that an engine will need during its life:

- Hot Section Inspection (low cost, 1-2 days downtime)
- Full Overhaul (high cost, 50-60 days downtime)

Most commercial jet engines have a **strict schedule** for Hot Section Interval (HIS) and Time Between Overhaul (TBO).

By modeling and predicting the failure events, we could transition to a **on-condition TBO cycle** that can result in reduced extended TBO and thus lower operating costs over the life of the engine.



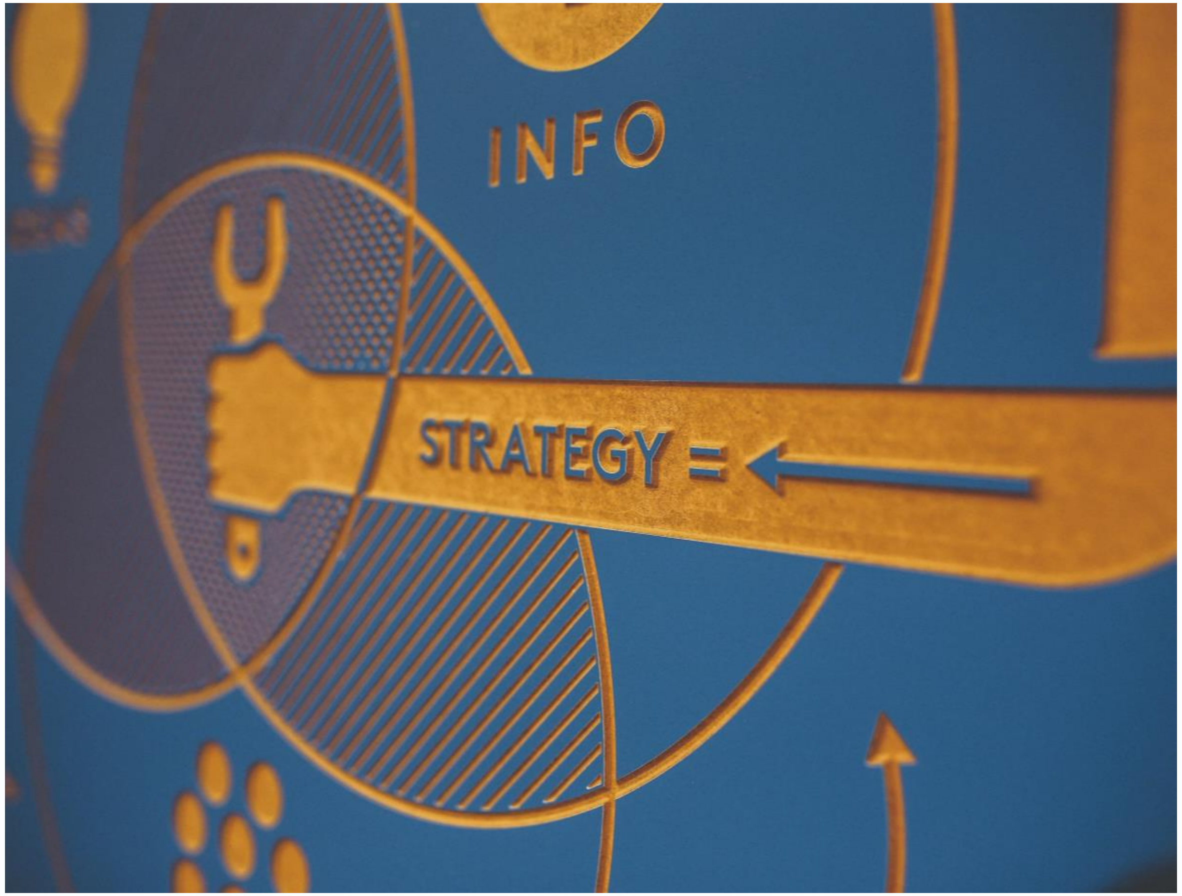
**Source:** "Jet Engine Maintenance: How Do You Manage Yours? | Avbuyer". 2020. Avbuyer.Com. https://www.avbuyer.com/articles/engines-biz-av/jet-engine-maintenance-how-do-you-manage-yours-69386.

# Value Creation

By modeling and predicting the failure events, we can transition from **schedule-based** to an **on-condition TBO cycle.** By safely extending the time in between maintenance events, we will lower the operating costs and increase the ROI of the engine.

Given the high cost of grounding a plane, extending the time in between service by only a few cycles will generate significant savings and thus the ROI for building this predictive model is very high.

Additional benefits include increased safety and more efficient logistics



# Dataset


To predict the imminent failure of an Engine we use simulated **run-to-failure** data provided by NASA. The dataset contains 4 datasets (FD001, FD002, ...) with different operating conditions and fault modes. It also includes a paper with the detail of how the data was generated.

```
data/raw
├── Damage Propagation Modeling.pdf
├── readme.txt
├── RUL_FD001.txt
├── RUL_FD002.txt
├── RUL_FD003.txt
├── RUL_FD004.txt
├── test_FD001.txt
├── test_FD002.txt
├── test_FD003.txt
├── test_FD004.txt
├── train_FD001.txt
├── train_FD002.txt
├── train_FD003.txt
└── train_FD004.txt
```



**Source:** A. Saxena and K. Goebel (2008). "Turbofan Engine Degradation Simulation Data Set", NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA

# Raw Data

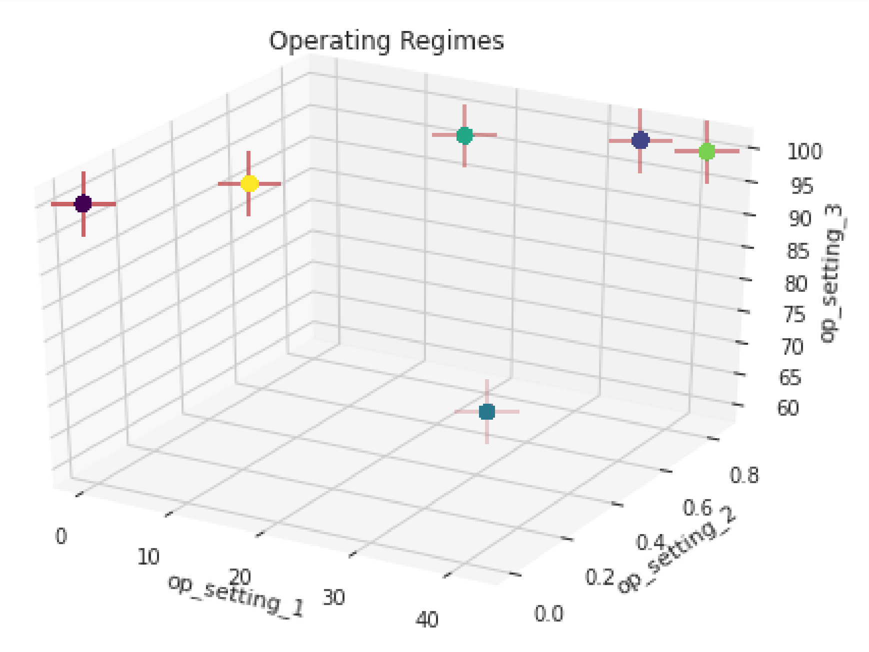
The data contains 3 operating settings and 21 sensor readings. The value for each sensor reading is the average value for each cycle. One cycle corresponds to one trip (take-off, flight, landing).

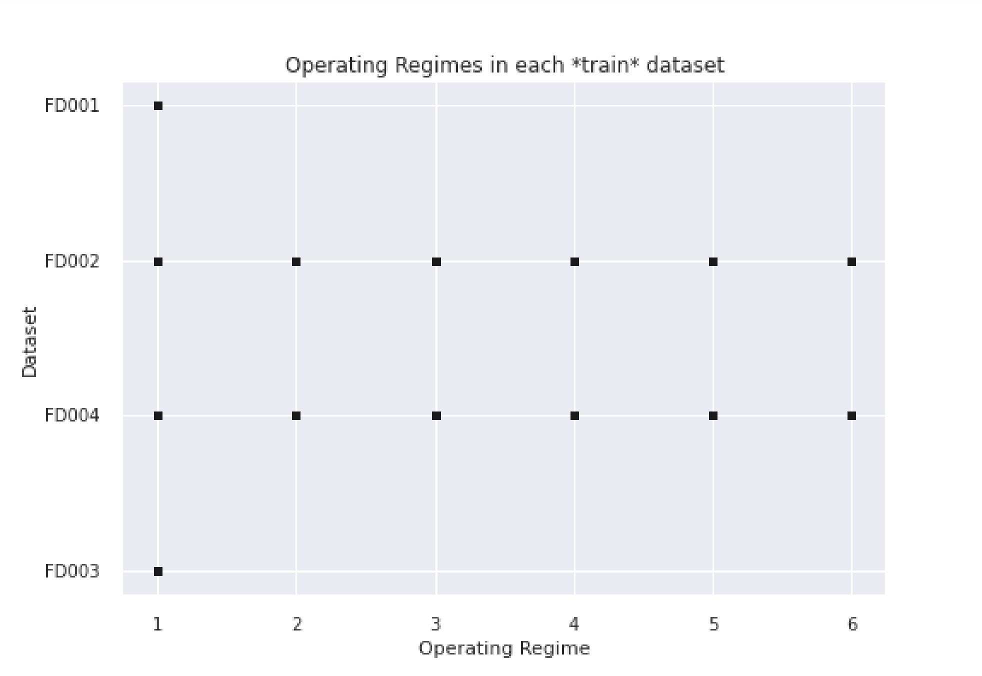
Being simulated data the dataset is very clean and required no maintenance (duplicates, missing values, outliers, etc.)

| | unit_number | cycle_time | op_setting_1 | op_setting_2 | op_setting_3 | s1 | s2 | s3 | s4 | s5 | ... | s21 | dataset |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | -0.0005 | 0.0004 | 100.0 | 518.67 | 642.36 | 1583.23 | 1396.84 | 14.62 | ... | 23.3537 | FD003 |
| **1** | 1 | 2 | 0.0008 | -0.0003 | 100.0 | 518.67 | 642.50 | 1584.69 | 1396.89 | 14.62 | ... | 23.4491 | FD003 |
| **2** | 1 | 3 | -0.0014 | -0.0002 | 100.0 | 518.67 | 642.18 | 1582.35 | 1405.61 | 14.62 | ... | 23.3669 | FD003 |
| **3** | 1 | 4 | -0.0020 | 0.0001 | 100.0 | 518.67 | 642.92 | 1585.61 | 1392.27 | 14.62 | ... | 23.2951 | FD003 |
| **4** | 1 | 5 | 0.0016 | 0.0000 | 100.0 | 518.67 | 641.68 | 1588.63 | 1397.65 | 14.62 | ... | 23.4583 | FD003 |

# Operating Regimes

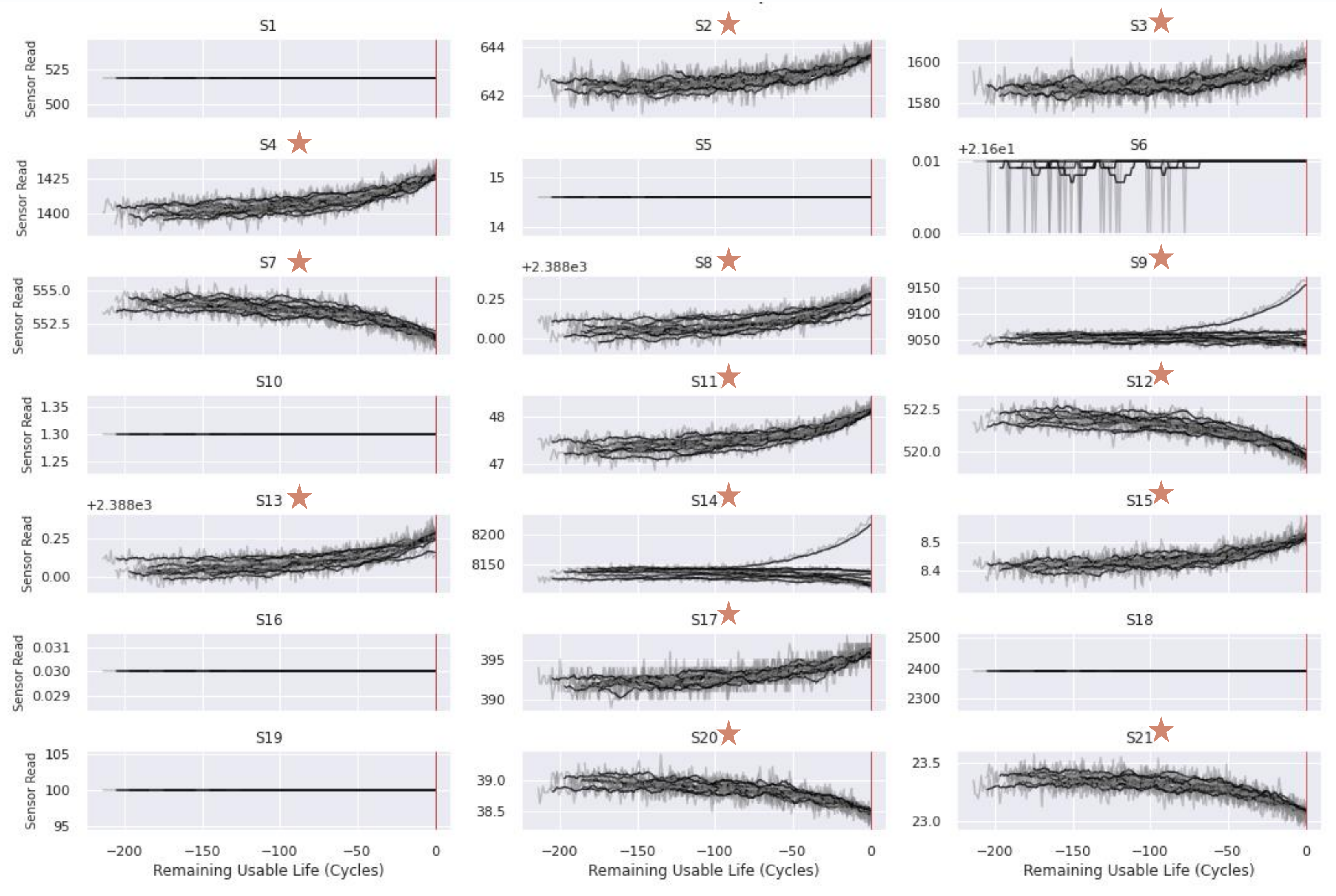
The data contains 3 operating settings, by visualizing these variables across different simuations we observed that they fall in to 6 distinct groups which we call **operating regimes.** We decide to narrow the scope of the prediction to a single operating condition. Thus we selected the dataset FD001 which displays a single operating regime.




Andrea Fantini

# Sensor Data



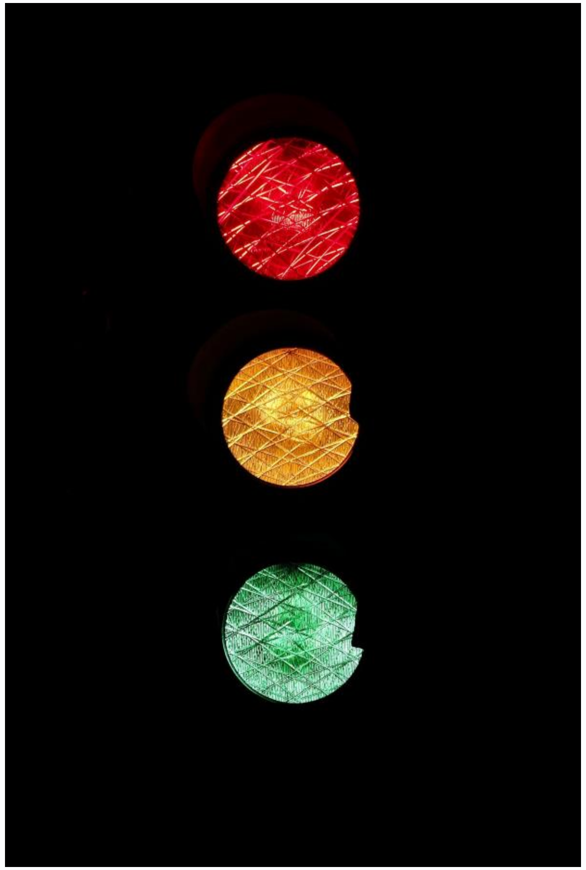

To poperly display the sensor traces we align them to the failure point which is the last data point in the series. We do not know how many cycles have occurred before the data collection has started.

We observe that seveal sensors have constant value throughout the dataset and others do not display any significant trends.

As we are planning to use a similarity model to make a predictions we select only the *starred* features to train our model.

Andrea Fantini

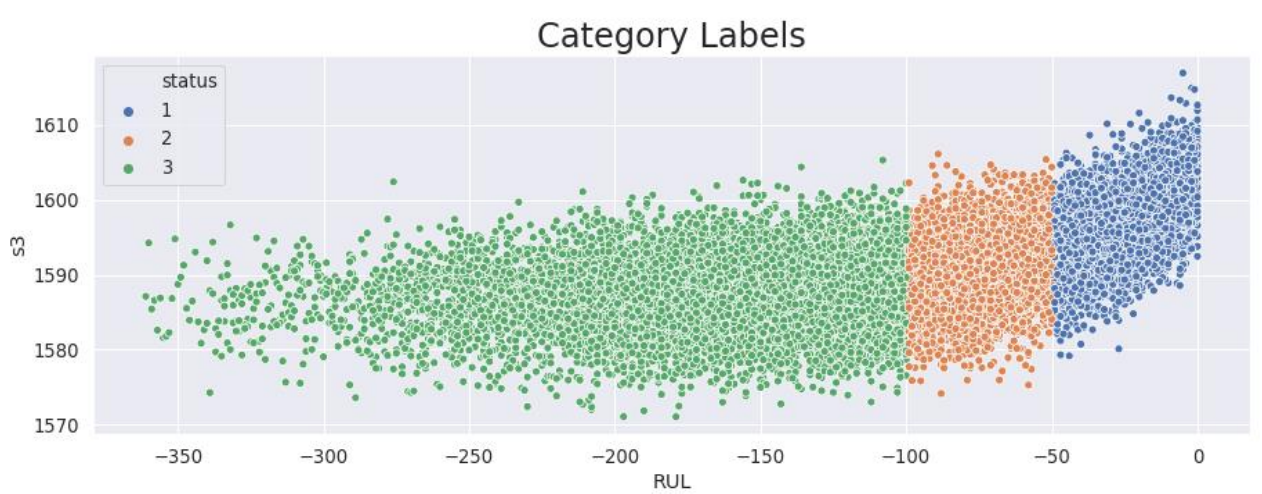
# Outcome Variable



We constructed an outcome variable based on the number of cycles before failure. This is likely to yield better predictions (as compared to predicting the number of cycles left) considering we don't know the starting condition of each engine. We decide to set two thresholds at 50 cycles and 100 cycles. These can be tuned to fit the scheduling needs*.

The goal is to predict the status of the engine and priority for service:

1. Alert
2. Warning
3. Normal

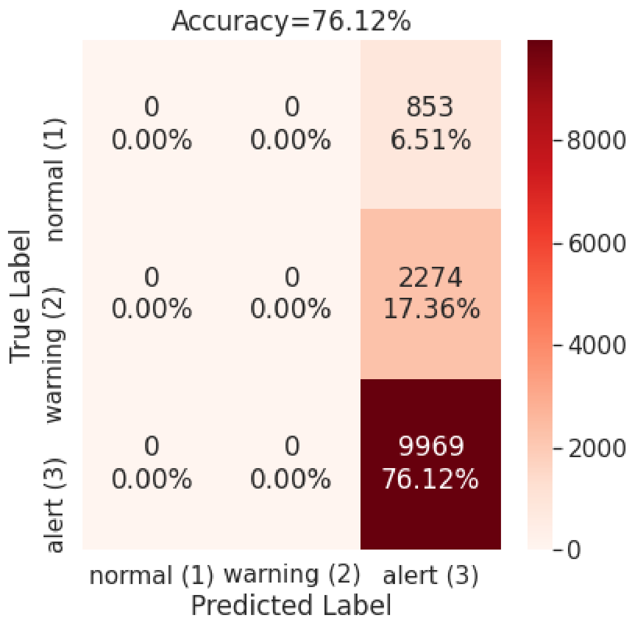



Andrea Fantini

*Tuning these parameters will affect the prediction performance

# Modeling - Baseline

To establish a baseline we use a Dummy Classifier which essentially will always generate and **Alert**. It's the most frequent class, statistically and intuitively it makes sense to air on the side of caution and always assume the engine will imminently fail. This obviously is not very useful, from a busines standpoint and could not replace a scheduled maintenance approach.




The resulting accuracy is ~76%

# Modeling – Classifier Comparison

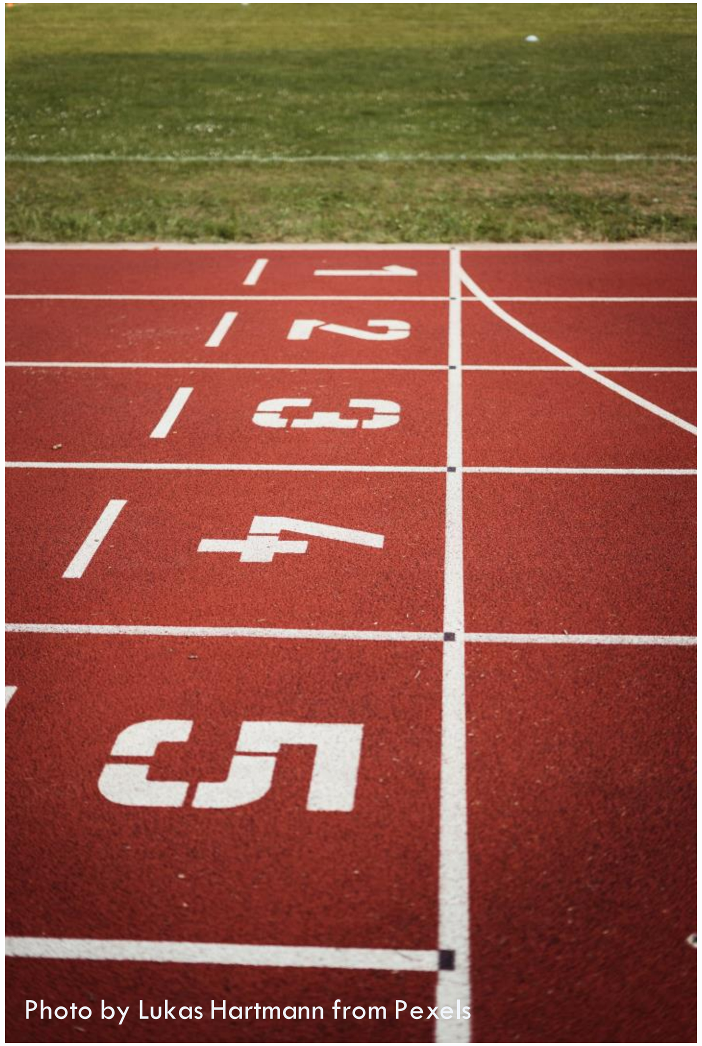
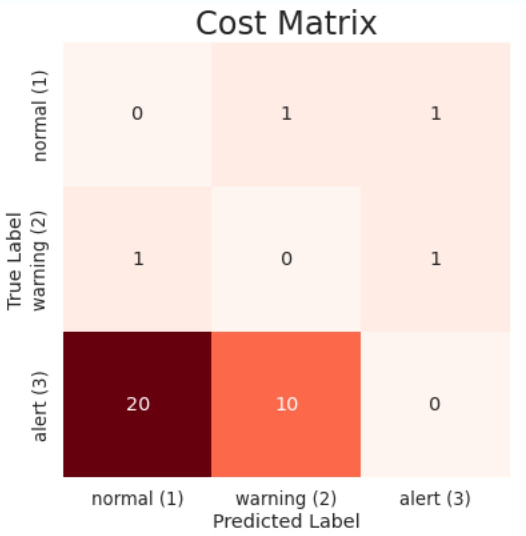


Photo by Lukas Hartmann from Pexels

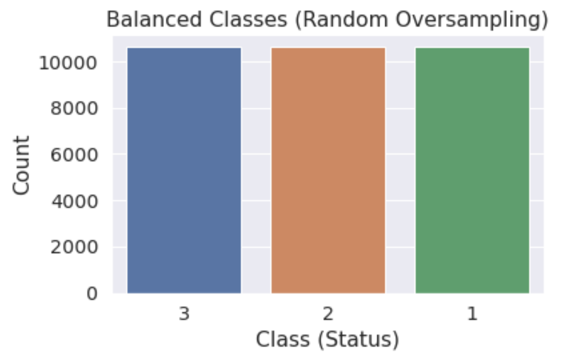
We compare the most common classifiers with the default parameters to select the best performers. Besides Accuracy we calculate the Cost, by defining a Cost Matrix:

- We attribute a high cost to **misleading** mistakes (engine is failing but the model predicts *normal* or *warning*) as they could lead to a failure during operation.
- We attribute a small cost to **conservative** mistakes (engine is *normal* but the model predicts an *alert* or *warning*) as they could lead to unnecessary maintenance.

Finally we use random oversampling to compensate for the class imbalance and see if that results in better performance.





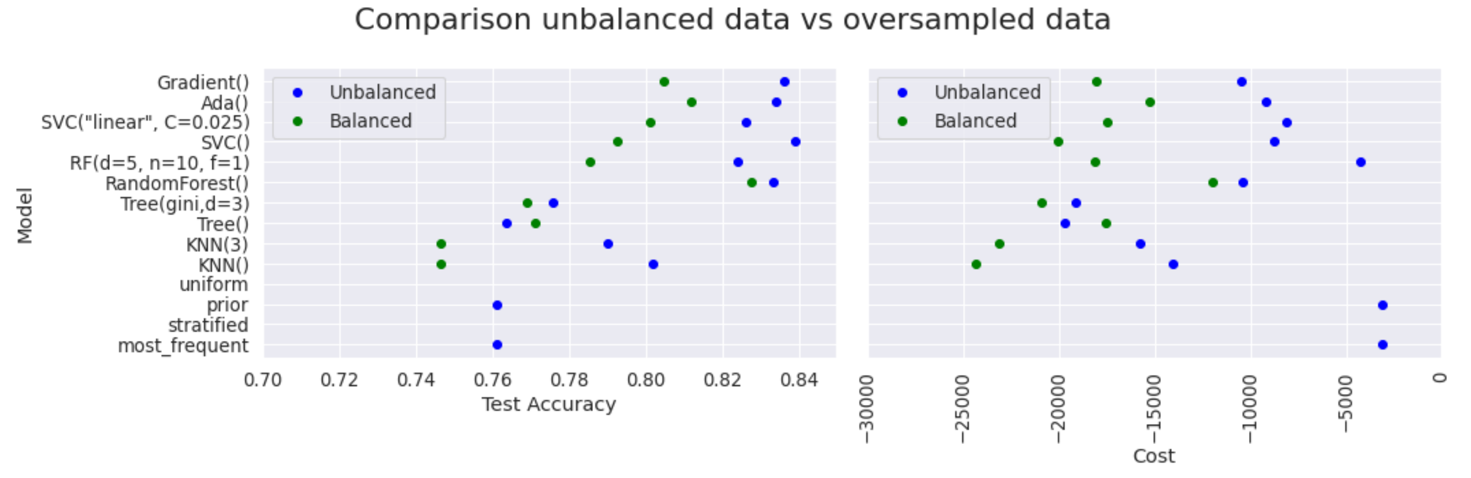



Andrea Fantini

# Modeling – Classifier Comparison Takeaways

- Balancing the classes has no positive effect on Accuracy or Cost. From now on we will work with the **original unbalanced data**
- The **Support Vector Machine** with rbf kernel has the highest Accuracy, and third lowest Cost
- The **Random Forest** model has the lowest cost and Accuracy in the top 6
- While the Baseline (*most_frequent*) Classifier has the lowest Cost it's impractical

| model_name | cost | test accuracy |
|---|---|---|
| SVC() | -8743.0 | 0.838958 ★ |
| GradientBoostingClassifier() | -10480.0 | 0.836210 |
| AdaBoostClassifier() | -9172.0 | 0.833919 |
| RandomForestClassifier() | -10382.0 | 0.833461 |
| SVC(C=0.025, kernel='linear') | -8091.0 | 0.826130 |
| RandomForestClassifier(max_depth=5, max_featur... | -4276.0 ★ | 0.824145 |
| KNeighborsClassifier() | -14083.0 | 0.801848 |
| KNeighborsClassifier(n_neighbors=3) | -15766.0 | 0.790012 |
| DecisionTreeClassifier(max_depth=3) | -19133.0 | 0.775733 |
| DecisionTreeClassifier() | -19718.0 | 0.763516 |
| DummyClassifier(strategy='most_frequent') | -3127.0 | 0.761225 |
| DummyClassifier(strategy='prior') | -3127.0 | 0.761225 |
| DummyClassifier(strategy='stratified') | -74684.0 | 0.449985 |
| DummyClassifier(strategy='uniform') | -101391.0 | 0.334835 |





Andrea Fantini

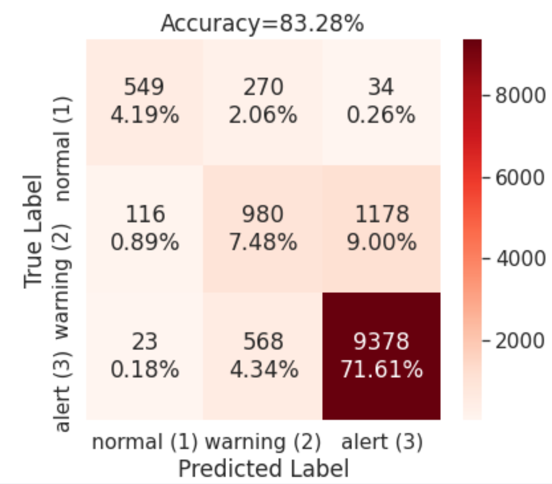
# Modeling – Hyperparameter Tuning

Photo by Karolina Grabowska from Pexels

The Random Forest Classifier is chosen for its lowest misclassification cost.

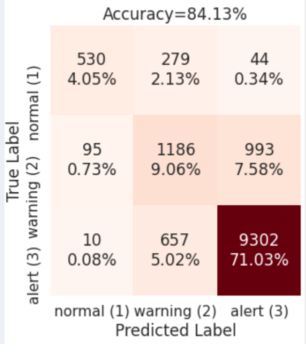
**Best model:**

```
pickle filename:
pickle_bayes_search_model_2020-10-18 20:03:06.122143.pkl

Classifier: RandomForest()

Model Parameters :
{('criterion', 'entropy'),
 ('max_depth', 6),
 ('n_estimators', 1000)}

CV Score :0.7919
Test Score :0.8328
Cost : -7738
```

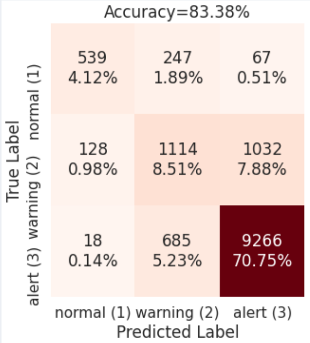



**Other models tuned:**

```
Classifier: SVC() rbf
Model Parameters : {('C', 8.846193496526725),
('gamma', 0.2547135012784452), ('kernel', 'rbf')}
Test Score :0.84135
Cost :-8181
```

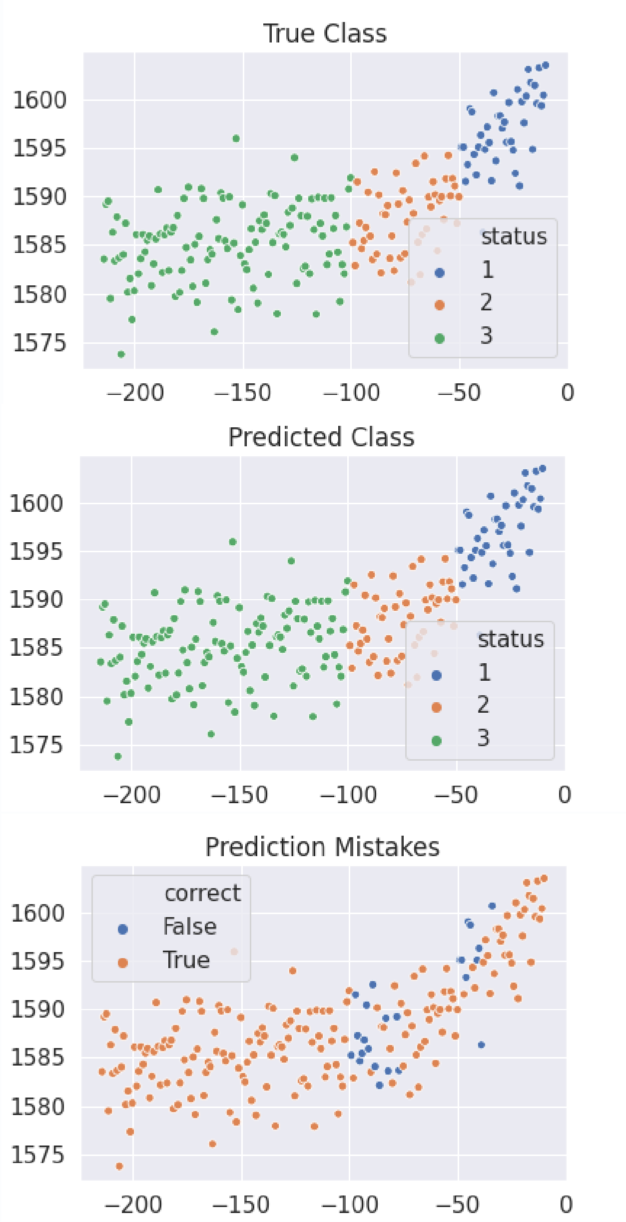



```
Classifier: SVC() linear
Model Parameters : {('C', 58873.32634924029),
('gamma', 60.64074528512966), ('kernel', 'linear')}
Test Score :0.8338
Cost :-8684
```



# Results

- Selecting a random sample from the test dataset we can simulate the collection of data points after each cycle.
- Unsurprisingly in this case the highest number of misclassification errors occurs at the boundary between two classes.
- Assuming this pattern is found to be representative of the majority of misclassification errors, it could be mitigated by choosing an appropriate maintenance policy



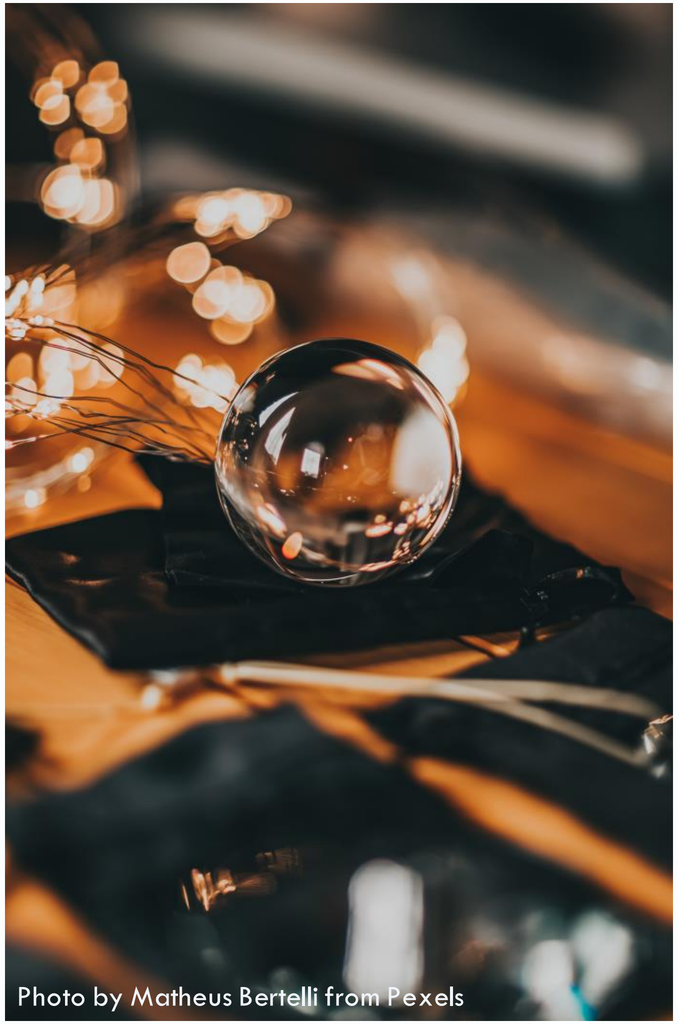



Photo by Matheus Bertelli from Pexels

Andrea Fantini

# Conclusions and Recommendations

The prediction model is able to predict engine failure with ~84% accuracy, in a limited set of operating conditions. While the nature of the data does not allow a direct estimation of the potential cost savings, the Cost matrix is a useful tool to identify the model with the lowest risk.

- Expand modelling to other operating regimes
- Explore ways to limit errors on the **Alert** category (use cost matrix as weights on the classification model)
- Simulate the effect of a three-strikes policy to mitigate misclassification errors (schedule urgent service after 3 warning alerts)
- Tweak class thresholds and cost matrix to align with business needs
- Simulate data from overhaul to failure to better quantify the cost savings derived from this model.

# Credits

Thanks to my mentor Devin Cavagnaro for the support and guidance in the development of this project.

Inspiration for this project came from the Predictive Maintenance with MATLAB A Prognostics Case Study

The technique to plot an overview of the sensors data was inspired from the blog post Predictive Maintenance for IoT by Ben Everson