Machine Learning for Predictive Maintenance: A Prognostic Model to Estimate End-of-Life of Turbofan Aircraft Engines

June 10, 2019

Contents

Introduction

This report will outline a method to predict remaining useful life (RUL) of commercial turbofan engines, based on a dataset containing values from multiple sensors and operational settings. The data set was provided by the Prognostics CoE at NASA Ames:

> The Modular Aero-Propulsion System Simulation (MAPSS) is a flexible turbofan engine simulation environment that provides easy access to health, control, and engine parameters.
>
> Both military and commercial turbofan engine versions of MAPSS exist. The commercial versions, C-MAPSS and C-MAPSS40k, represent high-bypass engines capable of 90,000 lbf thrust and 40,000 lbf thrust, respectively.

> Data sets consists of multiple multivariate time series. Each data set is further divided into training and test subsets. Each time series is from a different engine – the data can be considered to be from a fleet of engines of the same type. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e., it is not considered a fault condition. There are three operational settings that have a substantial effect on engine performance. These settings are also included in the data. The data is contaminated with sensor noise.
>
> Also provided [is] a vector of true Remaining Useful Life (RUL) values for the test data.

RUL is the target variable. Instead of treating this as a regression problem and estimating for RUL directly, this project will define an engine health status and classify engines as operating in one of three statuses: "Normal", "Monitor" or "Critical". Inspection or maintenance activities would be advisable for any engine that reaches the "Monitor" status.

Methods/Analysis

This analysis will be performed on the *FD001* test and train sets, and the associated vector of RUL values for the test set.

Exploratory Data Analysis

There are no null values in the provided Test, Train or RUL datasets.

Train	Test	RUL
FALSE	FALSE	FALSE

The training data appears as follows.

unit	cycle	setting1	setting2	setting3	sensor1	sensor2	sensor3	...	sensor21
1	1	-0.0007	-0.0004	100	518.67	641.82	1589.7	1400.6	23.419
1	2	0.0019	-0.0003	100	518.67	642.15	1591.82	1403.14	23.4236
1	3	-0.0043	0.0003	100	518.67	642.35	1587.99	1404.2	23.3442
1	4	0.0007	0	100	518.67	642.35	1582.79	1401.87	23.3739
1	5	-0.0019	-0.0002	100	518.67	642.37	1582.85	1406.22	23.4044
...	...	...	...	...	...	...	...	...	...
100	196	-0.0004	-0.0003	100	518.67	643.49	1597.98	1428.63	22.9735
100	197	-0.0016	-0.0005	100	518.67	643.54	1604.5	1433.58	23.1594
100	198	0.0004	0	100	518.67	643.42	1602.46	1428.18	22.9333
100	199	-0.0011	0.0003	100	518.67	643.23	1605.26	1426.53	23.064
100	200	-0.0032	-0.0005	100	518.67	643.85	1600.38	1432.14	23.0522

NASA, defines the "remaining useful life" for each engine as below:

> The engine is operating normally at the start of each time series, and develops a fault at some point during the series. In the training set, the fault grows in magnitude until system failure. In the test set, the time series ends some time prior to system failure.

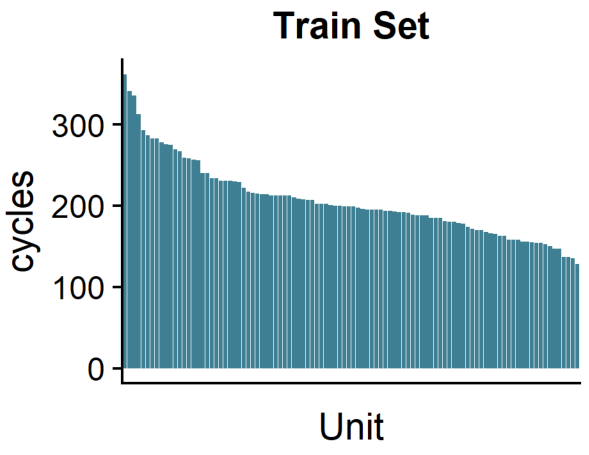
Therefore, for the training set, "remaining useful life" will be calculated for each engine, ending at 0 on the final cycle.

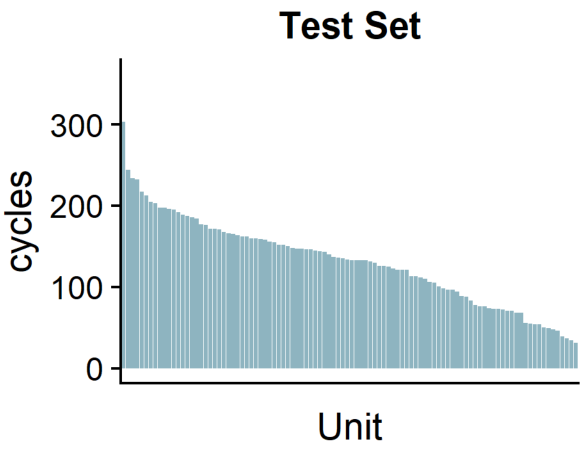
unit	cycle	setting1	setting2	setting3	sensor1	...	RUL
1	1	-0.0007	-0.0004	100	518.67	...	191
1	2	0.0019	-0.0003	100	518.67	...	190
1	3	-0.0043	0.0003	100	518.67	...	189
1	4	0.0007	0	100	518.67	...	188
1	5	-0.0019	-0.0002	100	518.67	...	187
...	...	...	...	...	...	...	...
1	190	-0.0027	0.0001	100	518.67	...	2
1	191	0	-0.0004	100	518.67	...	1
1	192	0.0009	0	100	518.67	...	0

For the test set, the "RUL" vector of values will be added to each engine on the final cycle, and then iteratively increased for each cycle prior. For example, Engine 1 had 112 cycles of "remaining useful life" at the point where the test data ends. With only 30 cycles recorded for Engine 1, it will therefore start with 142 cycles of "remaining useful life".

unit	cycle	setting1	setting2	setting3	sensor1	...	RUL
1	1	0.0023	0.0003	100	518.67	643.02	142
1	2	-0.0027	-0.0003	100	518.67	641.71	141
1	3	0.0003	0.0001	100	518.67	642.46	140
1	4	0.0042	0	100	518.67	642.44	139
1	5	0.0014	0	100	518.67	642.51	138
...	...	...	...	...	...	...	...
1	30	-0.0025	0.0004	100	518.67	642.79	113
1	31	-0.0006	0.0004	100	518.67	642.58	112

There are 100 engines in each dataset. As the chart below indicates, the *Test* engines contain fewer operating cycles.

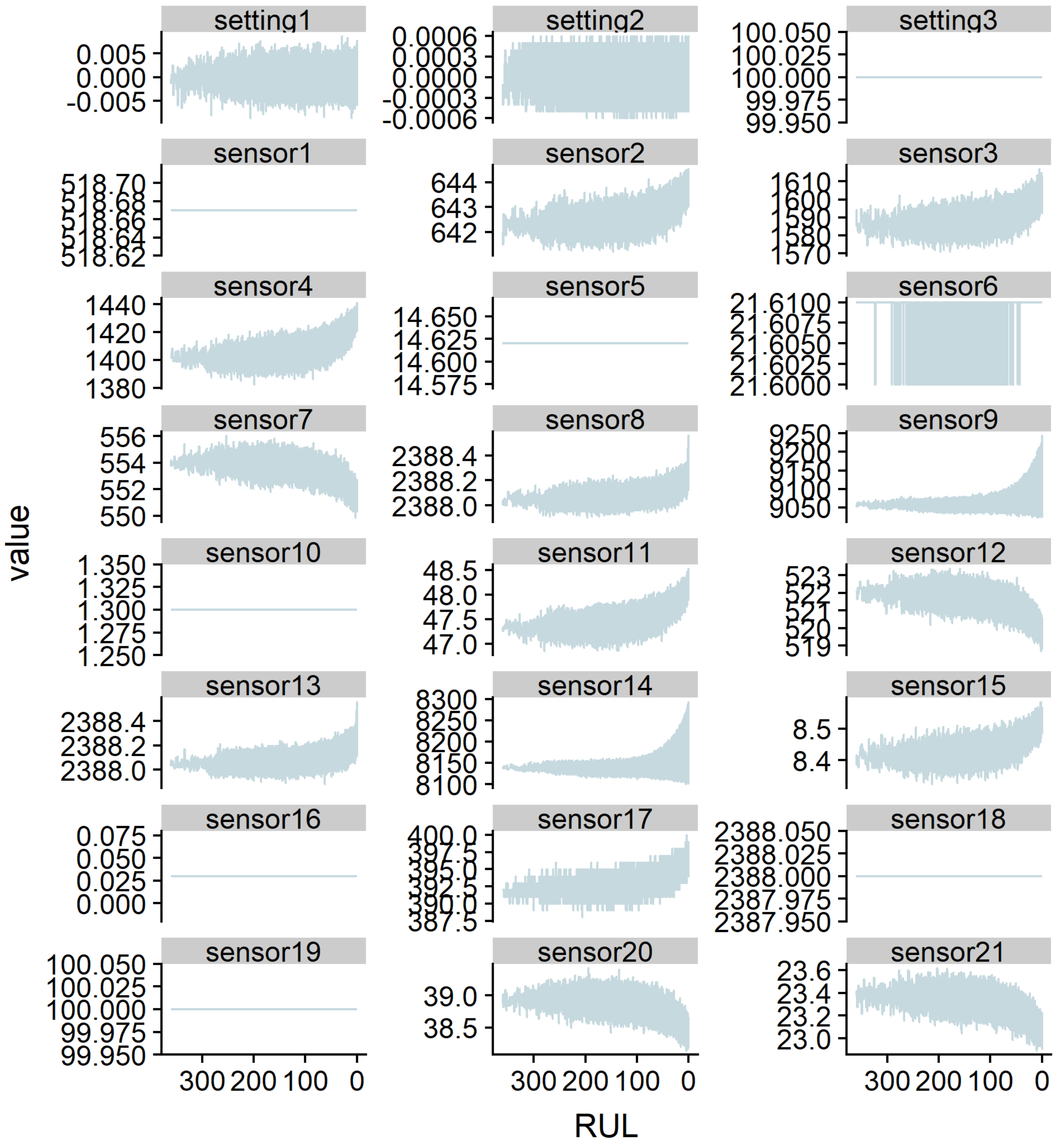




The test set is put away until it is used later to validate the model.

Sensor Visualization

The next figure displays a visual exploration of the sensor and settings values for each predictor in every engine of the training set. The chart is aligned "to the right" on the RUL values. In the training set, each engine is run until failure, and the right-most point on the x-axis is "RUL = 0" - the point of failure. This visualization accounts for the fact that not all engines have an equal number of cycles. Some sensors show a clear trend as the engine approaches failure.



Normalization / Standardization

The previous figure demonstrated clearly that the predictor values are not normalized, and some contain no meaningful content. The values for each setting and sensor will be pre-processed using three methods:

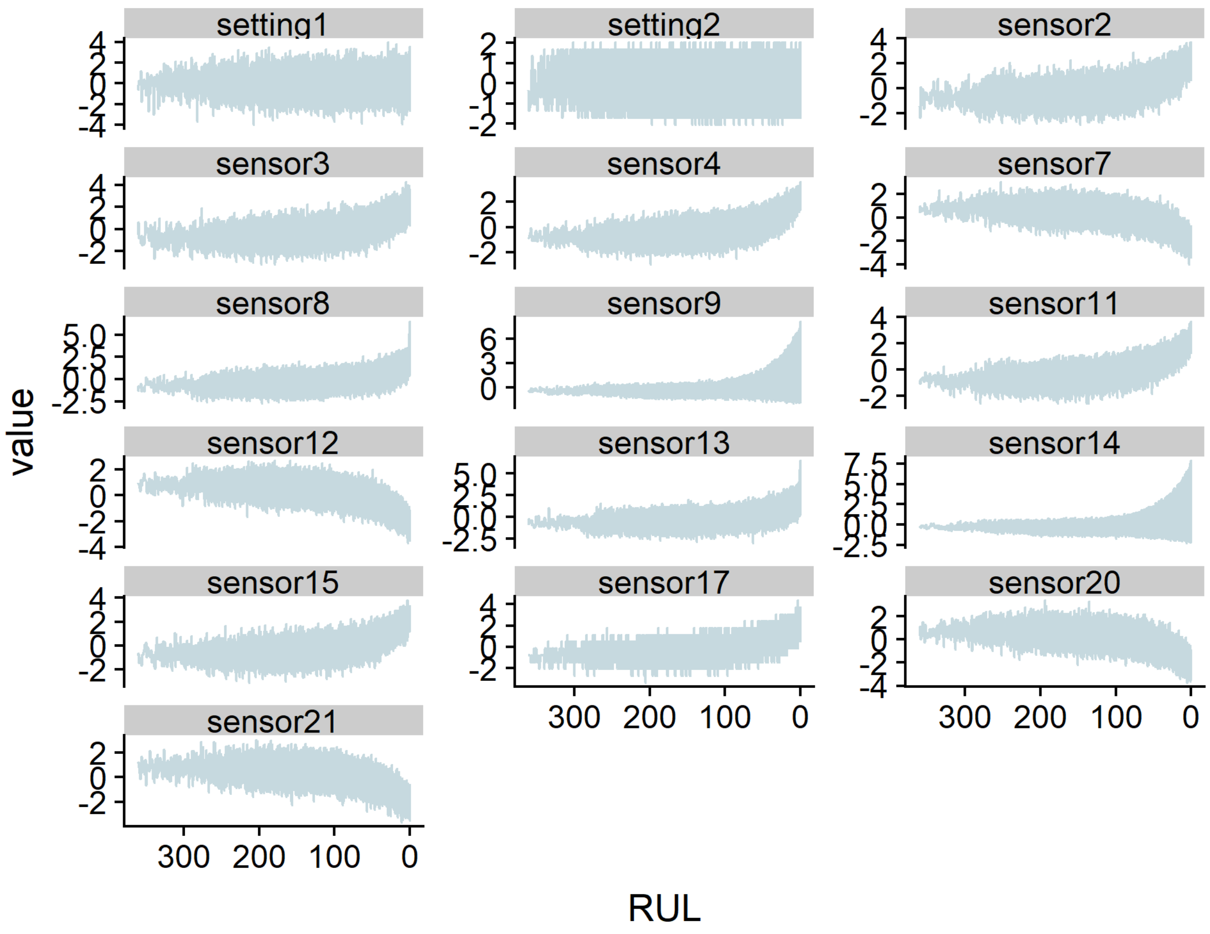
- *Scaling* the data to an interval between 0 and 1, by dividing the values by the standard deviation, and
- *Centering* the values, by subtracting the mean, and
- Removing predictors with *near-zero variance*

The below output shows which predictors were centered, scaled, and removed.

```
## $center
##  [1] "setting1" "setting2" "sensor2"  "sensor3"  "sensor4"  "sensor7"
##  [7] "sensor8"  "sensor9"  "sensor11" "sensor12" "sensor13" "sensor14"
## [13] "sensor15" "sensor17" "sensor20" "sensor21"
##
## $scale
##  [1] "setting1" "setting2" "sensor2"  "sensor3"  "sensor4"  "sensor7"
##  [7] "sensor8"  "sensor9"  "sensor11" "sensor12" "sensor13" "sensor14"
## [13] "sensor15" "sensor17" "sensor20" "sensor21"
##
## $ignore
## character(0)
##
## $remove
## [1] "setting3" "sensor1"  "sensor5"  "sensor6"  "sensor10" "sensor16"
## [7] "sensor18" "sensor19"
```

The same parameters used in pre-processing the training set will be applied to pre-processing the test set.

As the next figure shows, the data is now normalized:



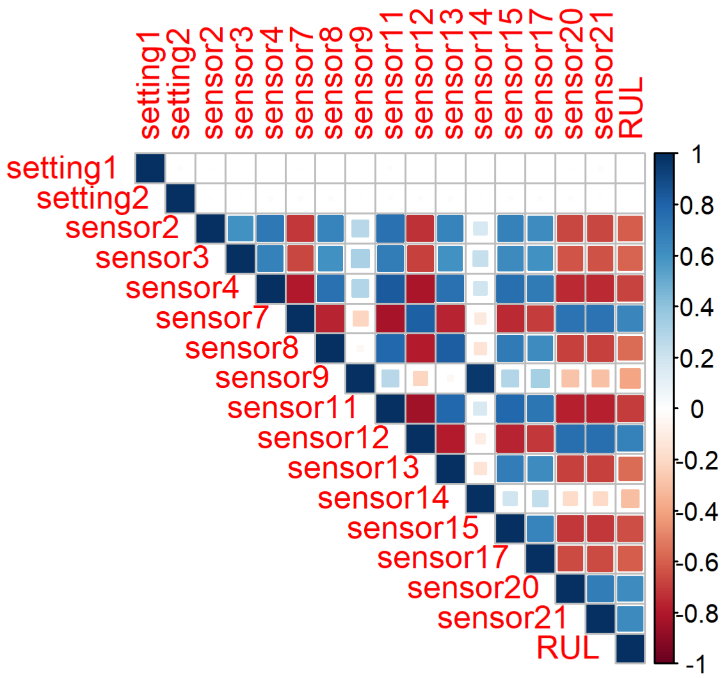
setting1
setting2
sensor2
sensor3
sensor4
sensor7
sensor8
sensor9
sensor11
sensor12
sensor13
sensor14
sensor15
sensor17
sensor20
sensor21
value
RUL
300
200
100
0

Predictor Correlation

A simple linear model is generated and then used to check for multicollinearity in the predictors, by computing the *Variance Inflation Factor*:

	x
unit	1.018397
cycle	2.127846
setting1	1.002198
setting2	1.001041
sensor2	2.640693
sensor3	2.292705
sensor4	4.601113
sensor7	4.345272
sensor8	4.714850
sensor9	17.891420
sensor11	6.033183
sensor12	5.302498
sensor13	4.701422
sensor14	17.281344
sensor15	3.303594
sensor17	2.564640
sensor20	3.070299
sensor21	3.118452

Sensors 9 and 14 have a VIF that is quite high relative to the rest of the data, indicating strong multicollinearity. Sensor 9 records the "Physical core speed (rpm)", whereas Sensor 14 records the "Corrected core speed (rpm)" (Saxena and Simon, 2008). "Corrected speed" adjusts component rotation to ambient conditions at sea level. A correlation plot may assist in determining which sensor to keep.



The remaining operational *settings* have very little correlation with RUL. Setting 1 (Altitude) and Setting 2 (Mach Number) will be removed from the data.

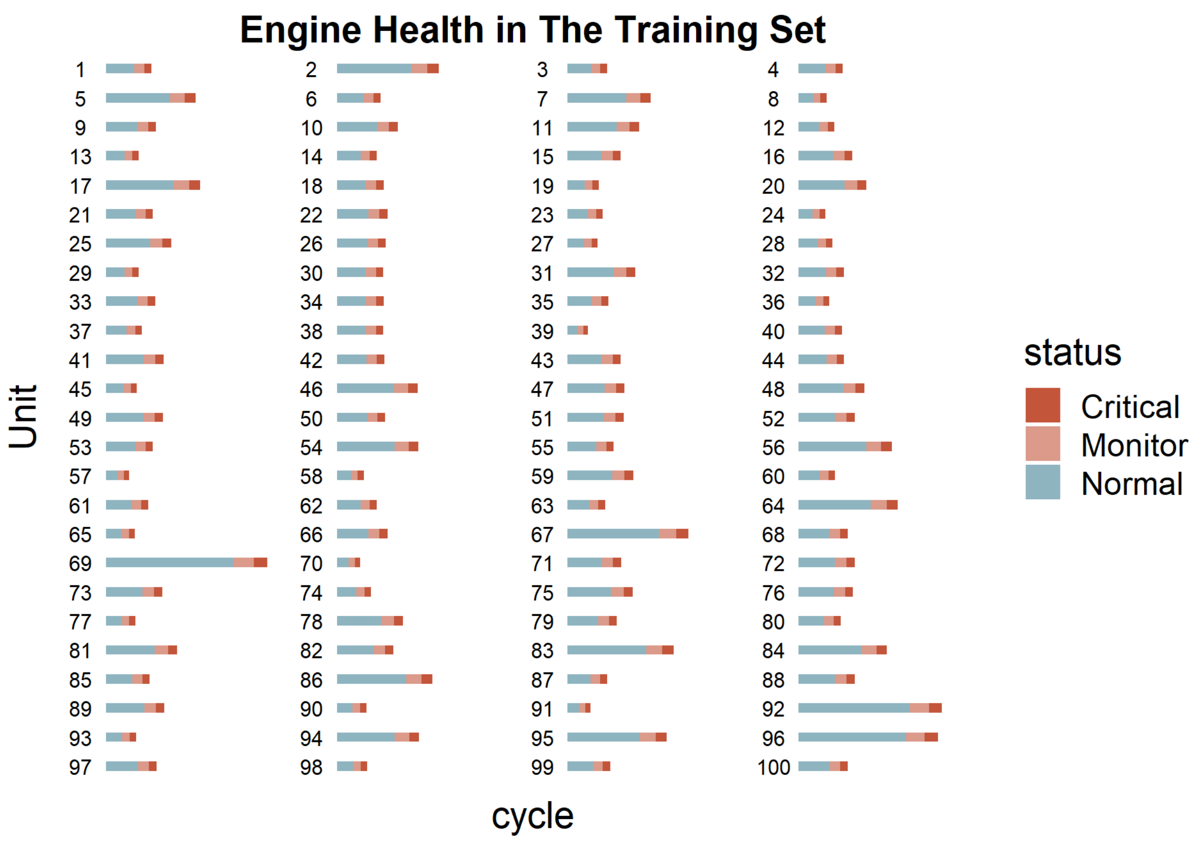
The remaining *sensors* are strongly correlated with each other, and most also have a strong correlation with RUL. Interestingly, Sensors 9 and 14 both have low correlation with the other sensors and with RUL. (As expected, they both have a strong correlation with each other.)

But Sensor 9 has a stronger correlation with RUL, so Sensor 14 will be dropped, and the VIF re-computed.

	x
unit	1.017572
cycle	2.124516
sensor2	2.640337
sensor3	2.290911
sensor4	4.600461
sensor7	4.343050
sensor8	4.660293
sensor9	2.061848
sensor11	6.030566
sensor12	5.292175
sensor13	4.651236
sensor15	3.303358
sensor17	2.563730
sensor20	3.070211
sensor21	3.116956

Feature Engineering

Three engine health statuses are added to the dataset. An engine will be flagged “Critical” when it has 15 or fewer cycles remaining. An engine will be flagged “Monitor” at 40 or fewer cycles, and “Normal” in all other cases.

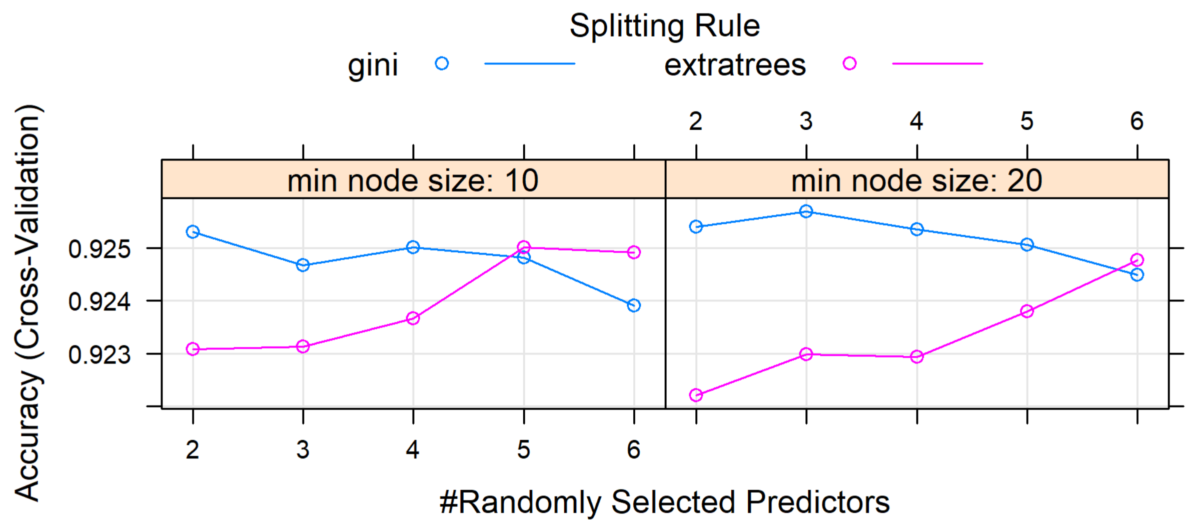


Modeling

A random forest was fit to the test data using the “ranger” method from the “caret” package. Cross-validation was completed using 5 folds.

Results

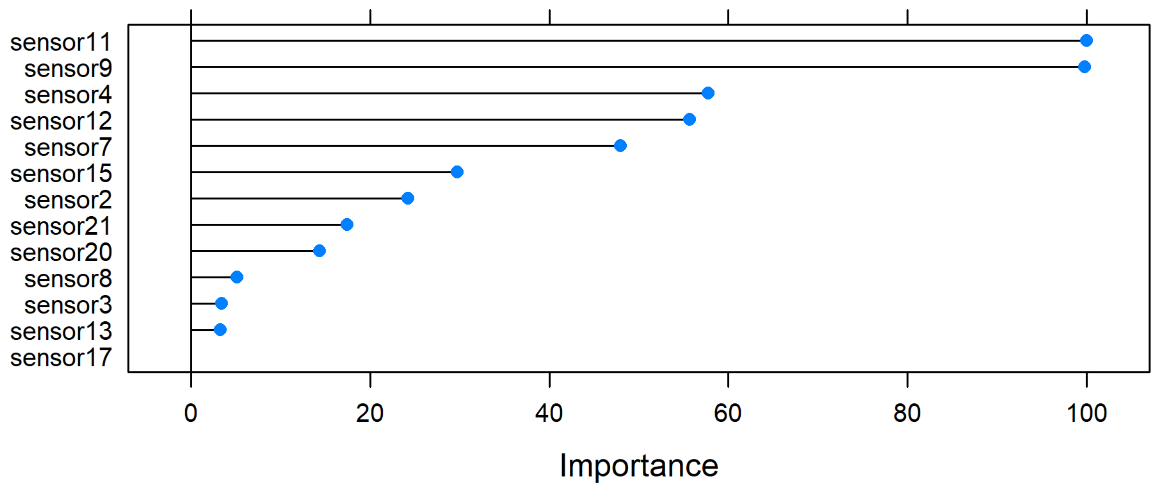
The following chart shows the combinations of “minimal node size”, “splitting rule”, and “mtry” (number of variables split at each node) used in fitting the model.



The best accuracy was achieved by choosing the `splitting rule` of gini, an `mtry` value of 3 and a node size of 20.

Variable Importance

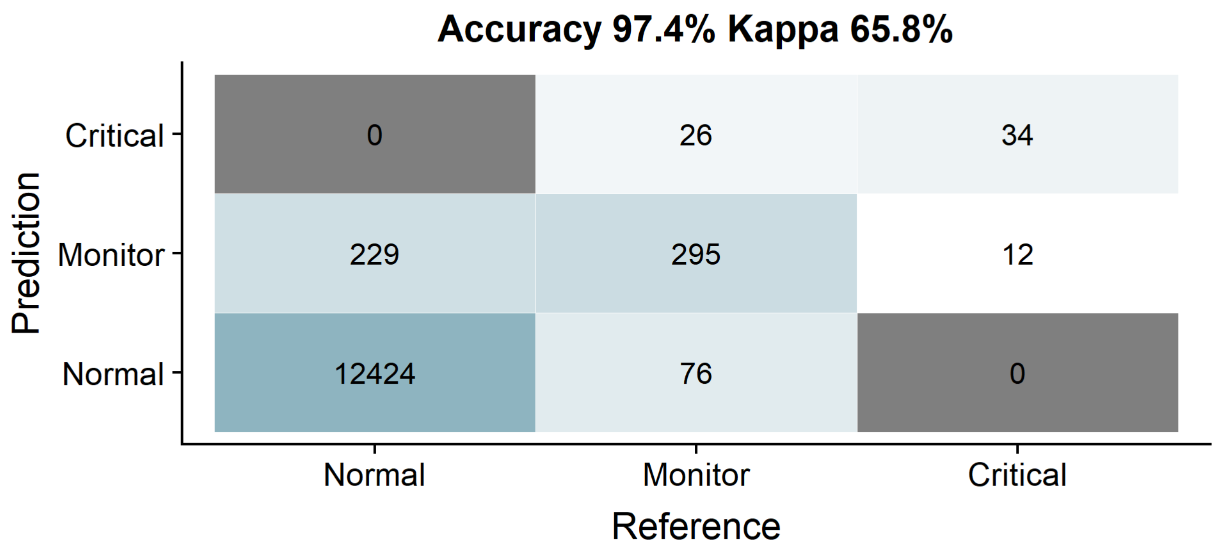
Based on the “variable importance plot”, the following predictors were most meaningful in the prediction.



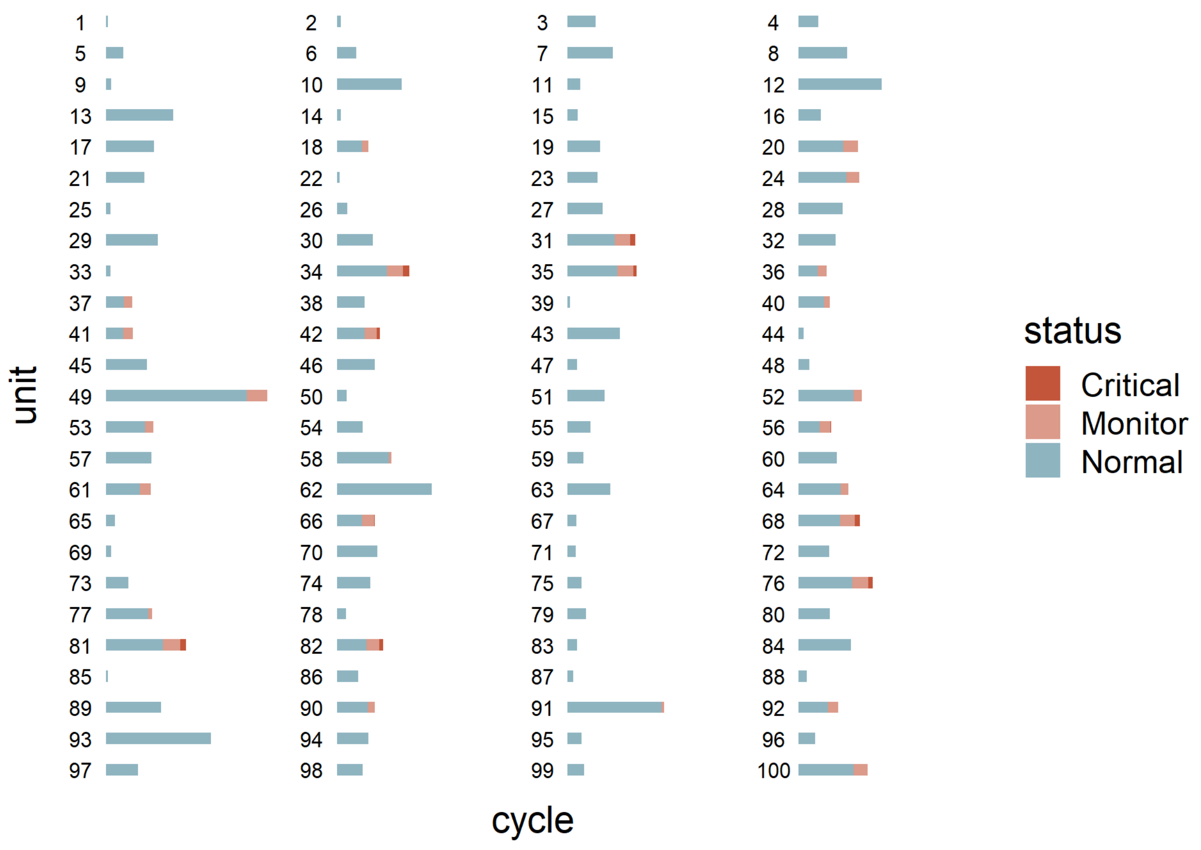
Confusion Matrix

The model achieved performance that is described by the following statistics.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Critical Monitor Normal
##   Critical       34      26      0
##   Monitor        12     295    229
##   Normal          0      76  12424
##
## Overall Statistics
##
##                Accuracy : 0.9738
##                  95% CI : (0.9709, 0.9765)
##     No Information Rate : 0.9662
##     P-Value [Acc > NIR] : 0.0000002956
##
##                   Kappa : 0.6578
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Critical Class: Monitor Class: Normal
## Sensitivity                 0.739130        0.74307        0.9819
## Specificity                 0.998008        0.98102        0.8284
## Pos Pred Value              0.566667        0.55037        0.9939
## Neg Pred Value              0.999079        0.99188        0.6158
## Prevalence                  0.003513        0.03031        0.9662
## Detection Rate              0.002596        0.02253        0.9487
## Detection Prevalence        0.004582        0.04093        0.9545
## Balanced Accuracy           0.868569        0.86205        0.9052
```



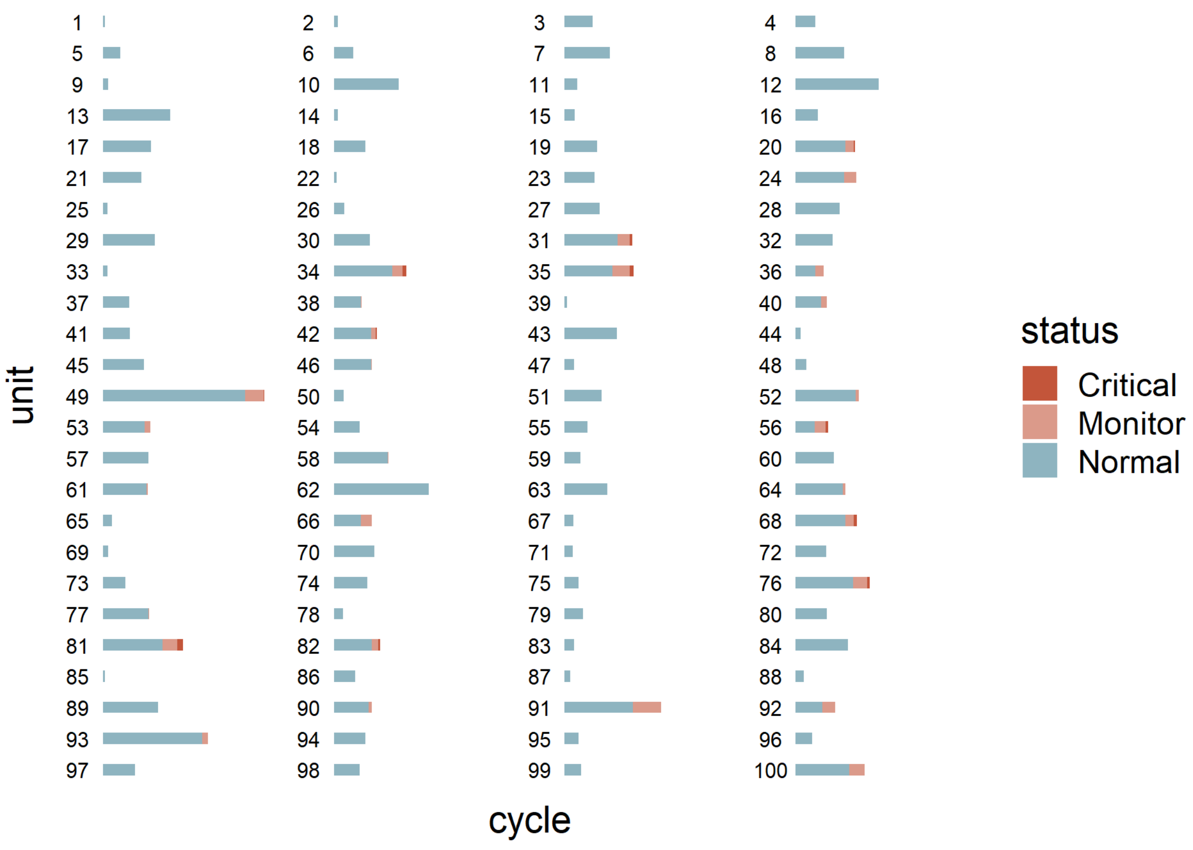
True Test Values



True vs. Predicted Values

The figures on this page and the next show the “true” vs. “predicted” statuses for each of the 100 engines in the test set. They can be compared by “flip booking” between the two pages.

Predicted Values



Conclusion

In the test set, no engine was run until failure. The engine that was closest to failure had only 7 cycles remaining.

The random forest model correctly classified the majority of engine states. When detecting faults in commercial turbofan aircraft engines, the worst kind of error is a False Negative - classifying an engine as free of fault conditions, when it is not. While False Positives may cause maintenance or inspection earlier than necessary, this is still preferable to an engine failing prematurely.

In the test set, all engines that were truly critical were classified in either the “Critical” or “Monitor” state, with the majority being correctly classified as “Critical”. Engines in the “monitor” state would ideally be inspected and/or maintained prior to reaching the “Critical” state or failing entirely.

Importantly, no engines in the “Critical” state were predicted to be in the “Normal” state, and likewise no engines in the “Normal” state were predicted to be in the “Critical” state.

Future Considerations

While not explored here, a neural network “Long-Short-Term Memory” (LSTM) approach may achieve improved performance, particularly with respect to improving sensitivity to the “Critical” status. Another option would be to treat this as a regression problem and try to estimate RUL directly. Methods of denoising or smoothing the data may also result in improvement.

Citations

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

A. Saxena and D. Simon (2008). “Damage propagation modeling for aircraft engine run-to-failure simulation”, International Conference on Prognostics and Health Management", October 6-9, 2008, Denver, CO.

A. Jain, P. Kundu and B. Lad (2014). “Prediction of Remaining Useful Life of an Aircraft Engine under Unknown Initial Wear”, 5th International & 26th All India Manufacturing Technology, Design and Research Conference (AIMTDR 2014)

“Modular Aero-Propulsion System Simulations - MAPSS, C-MAPSS, C-MAPSS40k”, Nasa Intelligent Control and Autonomy Branch (https://www.grc.nasa.gov/www/cdtb/software/mapss.html), NASA Glenn Research Center