Machine Learning Engineer Nanodegree

Capstone Project - Predictive Maintenance

Ross Green - 17 April 2018

I. Definition

Project Overview

Unscheduled equipment downtime can be extremely detrimental for business and can have affects which ripple across the entire supply chain. The ability to identify issues that can lead to failures and to have an accurate view of the assets remaining lifetime could have huge cost savings and maximise utilisation and performance for a company.

Real time sensor data from networked industrial machines such as a fleet of trucks allows enables real time diagnosis of problems and as-well-as predictions on the future health of individual units which can enable pro-active maintenance, as opposed to reactive or break-fix maintenance. Predictive maintenance is an exciting field and frequently cited as one of the most promising and lucrative industrial applications of the IoT.

This project explores a predictive maintenance solution on Turbofan Engine Degradation Simulation data set which was released in 2008 by the Prognostics Center of Excellence at NASA's Ames research center. It consists of sensor readings from a fleet of simulated aircraft gas turbine engines, recorded as multiple multivariate time series.

By monitoring time series operational data about an asset, patterns in the degradation of the asset over its lifetime can be found. A machine learning approach is used to find these patterns and use them to predict failure conditions of an in-service asset in order to inform actionable outcomes to optimise maintenance and use of the asset to achieve business goals.

This project is based on the Predictive Maintenance Expirement on Azure ML.

Problem Statement

The problem statement is stated as "Given an assets operational and failure events history, can I predict when an in-service asset will fail?"

This can be formulated as the following outcomes:

Predicting the Remaining Useful Life (RUL) of the asset. This is predicting how many more cycles an in-service asset will run before it fails. Since this is a continuous variable this is a regression problem.

Predicting if an asset will fail within a certain timeframe. For example, will it fail within the next 30 cycles. Since this is predicting between 2 classes (will fail in 30 cycles, wont fail within 30 cycles) this is a binary classification problem.

Predicting if an asset will fail within different time windows. For example, will it fail within the next 15 cycles, will it fail within the next 15 to 30 cycles, or will it fail after 30 cycles. This is a multiclass classification problem.

After defining and creating the above target variables as RUL, label1 and label2, a cost metric is defined that captures the penalty associated with incorrect predictions which is

used to compare the performance of different models. Simple linear regression on the raw input sensor data is used to set a benchmark level of performance, then after an exploratory analysis of the dataset and some feature engineering, more complex models including Random Forrests and LSTM Neural Networks are used to capture more of the complexities in the data. The models are evaluated using Cross Validation and hyper-parameters tuned to give the best performing model.

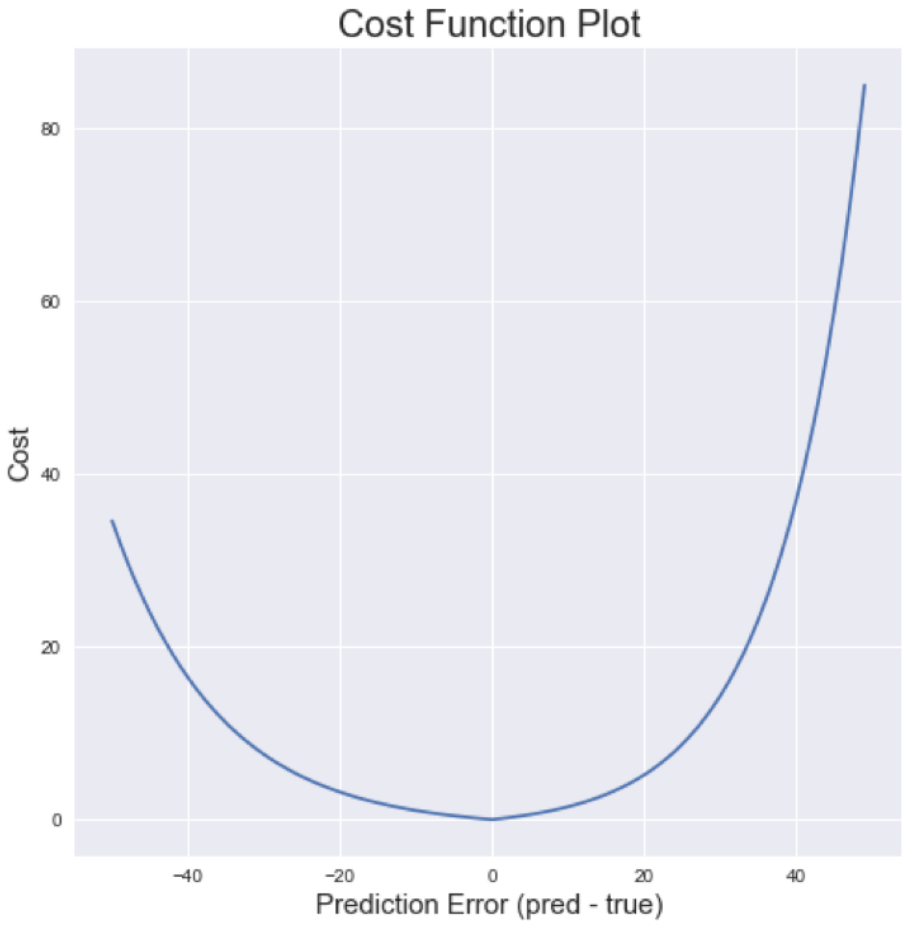
Metrics

In determining the metrics to use, the cost of an incorrect prediction must be considered. In the case of predictive maintenance, we consider that a model is trained on the above data and used in production to tell us when to service an engine. If the model underestimates the RUL of the engine, then it would be serviced earlier than it otherwise could have been since it could have operated without issue for longer. On the other hand, if the model overestimates the RUL of an engine, then it could potentially fail during operation when we thought there would be more time which could be much worse consequences.

To capture the cost of different incorrect predictions I use the following cost function:

$$Cost = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} e^{-\frac{pred-actual}{14}} - 1, & \text{if } pred - actual < 0 \\ \frac{1}{N}\sum_{i=1}^{N} e^{\frac{pred-actual}{11}} - 1, & \text{if } pred - actual \geq 0 \end{cases}$$

which gives the plot below:



The cost for the label predictions is similar to the Accuracy score accept that the cost for a prediction which is less than the actual is double the cost otherwise. This captures the fact that predicting an asset is not about to fail with a time window (1 for label 1, 1 or 2 for label 2) when it actually will fail is worse than predicting that it will fail when it actually wont.

$$Cost = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} 2 * (pred - actual), & \text{if } pred - actual < 0 \\ \frac{1}{N}\sum_{i=1}^{N} pred - actual, & \text{if } pred - actual \geq 0 \end{cases}$$

II. Analysis

Data Exploration

The datasets contain sensor readings from a fleet of simulated aircraft gas turbine engines recorded as multiple multivariate time series where 'cycle' is the time unit.

Column	Description
dataset_id	id of the original dataset of this instance
unit_id	id of engine (unique in each dataset)
cycle	number of operational cycles since beginning of engine operation
setting 1	value of operational setting 1
setting 2	value of operational setting 2
setting 3	value of operational setting 3
sensor 1	value of sensor 1
...	...
sensor 21	value of sensor 21

Sample training data

~20k rows,
100 unique engine id

id	cycle	setting1	setting2	setting3	s1	s2	s3	...	s19	s20	s21
1	1	-0.0007	-0.0004	100	518.67	641.82	1589.7		100	39.06	23.419
1	2	0.0019	-0.0003	100	518.67	642.15	1591.82		100	39	23.4236
1	3	-0.0043	0.0003	100	518.67	642.35	1587.99		100	38.95	23.3442
...	...										
1	191	0	-0.0004	100	518.67	643.34	1602.36		100	38.45	23.1295
1	192	0.0009	0	100	518.67	643.54	1601.41		100	38.48	22.9649
2	1	-0.0018	0.0006	100	518.67	641.89	1583.84		100	38.94	23.4585
2	2	0.0043	-0.0003	100	518.67	641.82	1587.05		100	39.06	23.4085
2	3	0.0018	0.0003	100	518.67	641.55	1588.32		100	39.11	23.425
...	...										
2	286	-0.001	-0.0003	100	518.67	643.44	1603.63		100	38.33	23.0169
2	287	-0.0005	0.0006	100	518.67	643.85	1608.5		100	38.43	23.0848

Sample testing data

~13k rows,
100 unique engine id

id	cycle	setting1	setting2	setting3	s1	s2	s3	...	s19	s20	s21
1	1	0.0023	0.0003	100	518.67	643.02	1585.29		100	38.86	23.3735
1	2	-0.0027	-0.0003	100	518.67	641.71	1588.45		100	39.02	23.3916
1	3	0.0003	0.0001	100	518.67	642.46	1586.94		100	39.08	23.4166
...	...										
1	30	-0.0025	0.0004	100	518.67	642.79	1585.72		100	39.09	23.4069
1	31	-0.0006	0.0004	100	518.67	642.58	1581.22		100	38.81	23.3552
2	1	-0.0009	0.0004	100	518.67	642.66	1589.3		100	39	23.3923
2	2	-0.0011	0.0002	100	518.67	642.51	1588.43		100	38.84	23.2902
2	3	0.0002	0.0003	100	518.67	642.58	1595.6		100	39.02	23.4064
...	...										
2	48	0.0011	-0.0001	100	518.67	642.64	1587.71		100	38.99	23.2918
2	49	0.0018	-0.0001	100	518.67	642.55	1586.59		100	38.81	23.2618
3	1	-0.0001	0.0001	100	518.67	642.03	1589.92		100	38.99	23.296
3	2	0.0039	-0.0003	100	518.67	642.23	1597.31		100	38.84	23.3191
3	3	0.0006	0.0003	100	518.67	642.98	1586.77		100	38.69	23.3774
...	...										
3	125	0.0014	0.0002	100	518.67	643.24	1588.64		100	38.56	23.227
3	126	-0.0016	0.0004	100	518.67	642.88	1589.75		100	38.93	23.274

Sample ground truth data

100 rows

RUL
112
98
69
82
91

Each sensor measures a specific physical attribute of the engine for example temperature, vibration, speed etc.

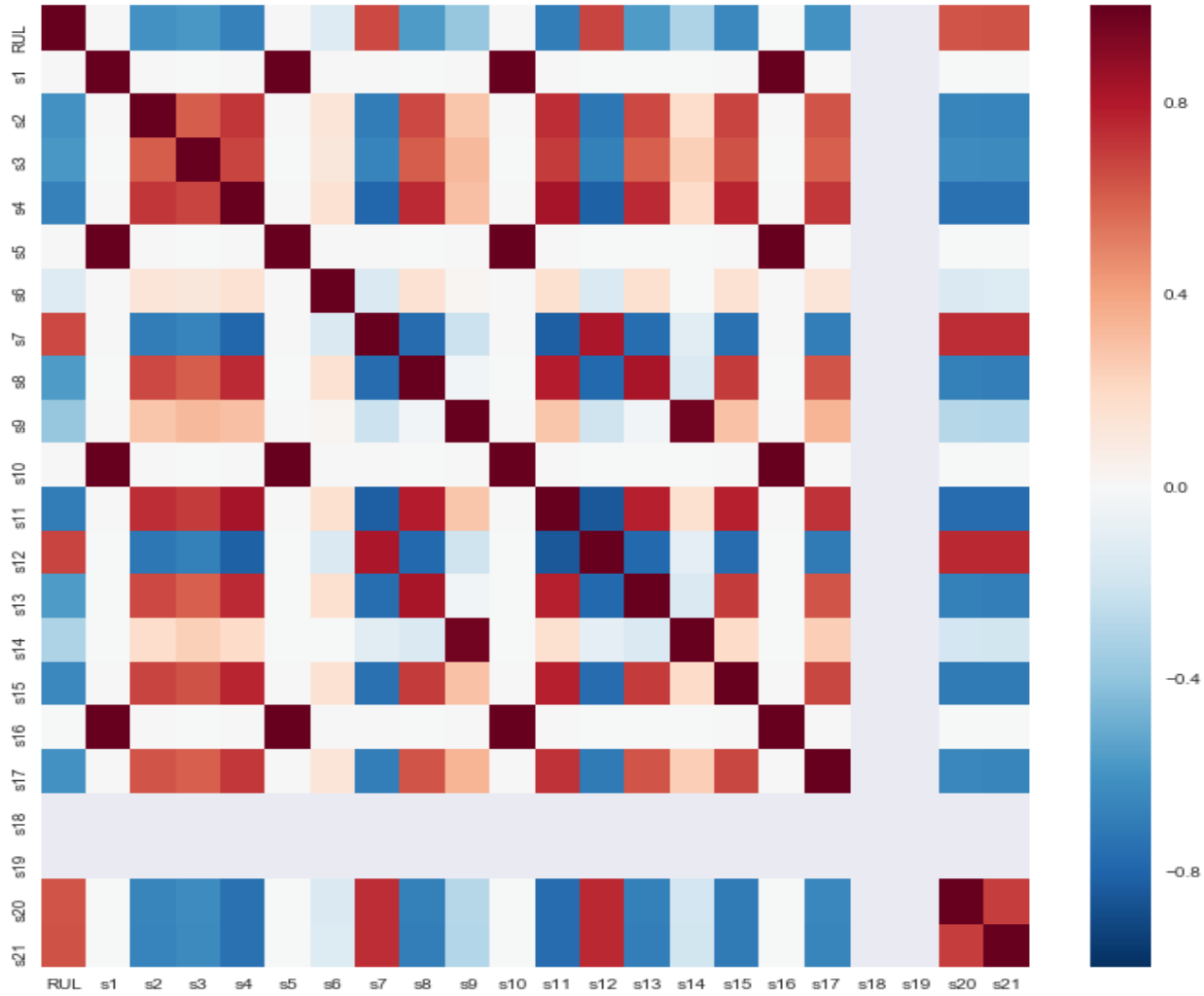
The training dataset provides data up until the the engine fails, so the last cycle is when the failure event occurs. The test data only provides a window of the assets operational lifetime and gives no indication of when the failure event occurs. The ground truth dataset provides the number of cycles remaining for the test set. For example, for engine with ID 1, we are given the operational data up until cycle 31, and the ground truth dataset shows that the engine will run for another 112 cycles before it fails.

Exploratory Visualisation

The min, max, mean, median, std, skew and kurtosis for each sensor value and the operational settings values are shown below.

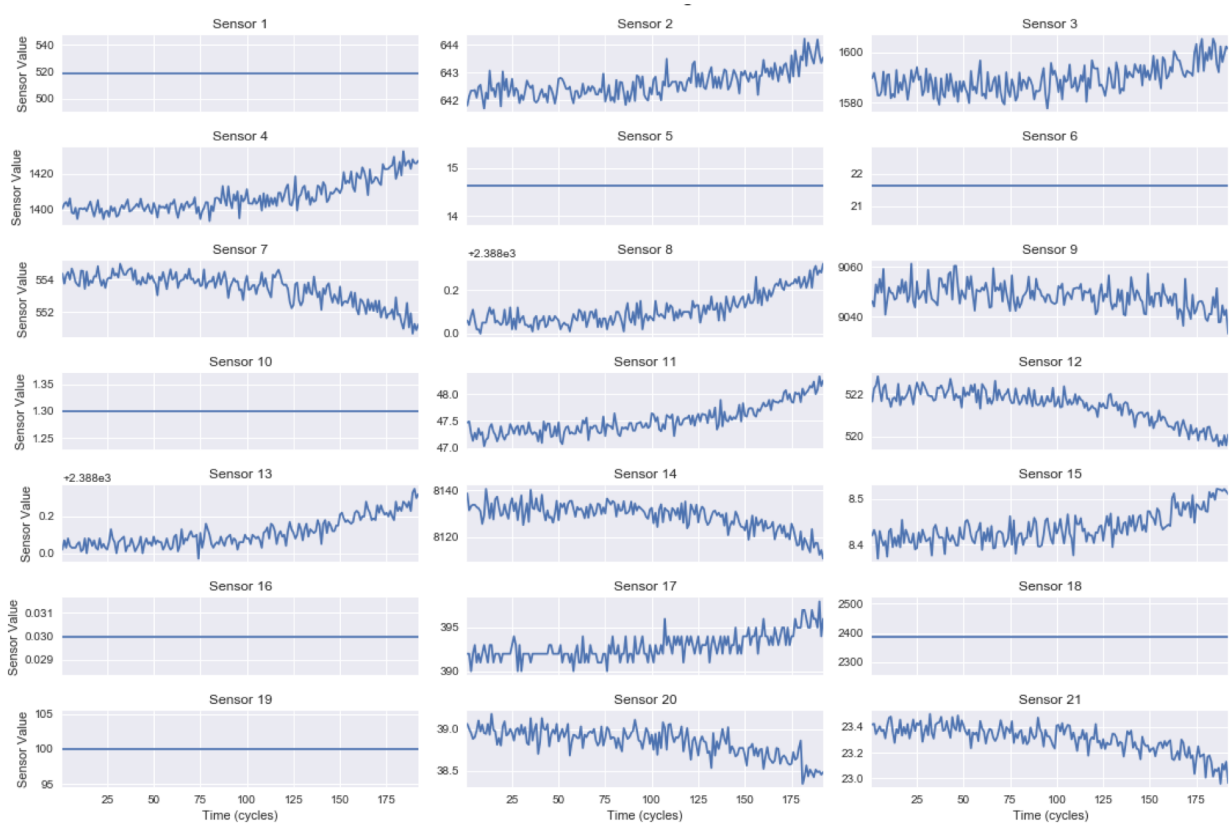
	min	max	mean	median	std	skew	kurtosis
s1	518.6700	518.6700	518.670000	518.6700	6.537152e-11	0.000000	0.000000
s2	641.2100	644.5300	642.680934	642.6400	5.000533e-01	0.316526	-0.112043
s3	1571.0400	1616.9100	1590.523119	1590.1000	6.131150e+00	0.308946	0.007762
s4	1382.2500	1441.4900	1408.933782	1408.0400	9.000605e+00	0.443194	-0.163681
s5	14.6200	14.6200	14.620000	14.6200	3.394700e-12	0.000000	0.000000
s6	21.6000	21.6100	21.609803	21.6100	1.388985e-03	-6.916813	45.846746
s7	549.8500	556.0600	553.367711	553.4400	8.850923e-01	-0.394329	-0.157949
s8	2387.9000	2388.5600	2388.096652	2388.0900	7.098548e-02	0.479411	0.333149
s9	9021.7300	9244.5900	9065.242941	9060.6600	2.208288e+01	2.555365	9.378681
s10	1.3000	1.3000	1.300000	1.3000	4.660829e-13	0.000000	0.000000
s11	46.8500	48.5300	47.541168	47.5100	2.670874e-01	0.469329	-0.172192
s12	518.6900	523.3800	521.413470	521.4800	7.375534e-01	-0.442407	-0.144917
s13	2387.8800	2388.5600	2388.096152	2388.0900	7.191892e-02	0.469792	0.387244
s14	8099.9400	8293.7200	8143.752722	8140.5400	1.907618e+01	2.372554	8.854664
s15	8.3249	8.5848	8.442146	8.4389	3.750504e-02	0.388259	-0.121430
s16	0.0300	0.0300	0.030000	0.0300	1.556432e-14	0.000000	0.000000
s17	388.0000	400.0000	393.210654	393.0000	1.548763e+00	0.353126	-0.039174
s18	2388.0000	2388.0000	2388.000000	2388.0000	0.000000e+00	0.000000	0.000000
s19	100.0000	100.0000	100.000000	100.0000	0.000000e+00	0.000000	0.000000
s20	38.1400	39.4300	38.816271	38.8300	1.807464e-01	-0.358445	-0.112829
s21	22.8942	23.6184	23.289705	23.2979	1.082509e-01	-0.350375	-0.117039
setting1	-0.0087	0.0087	-0.000009	-0.0000	2.187313e-03	-0.024766	-0.009132
setting2	-0.0006	0.0006	0.000002	0.0000	2.930621e-04	0.009085	-1.130447
setting3	100.0000	100.0000	100.000000	100.0000	0.000000e+00	0.000000	0.000000

The heat map below shows the correlation between the sensor values and the RUL.

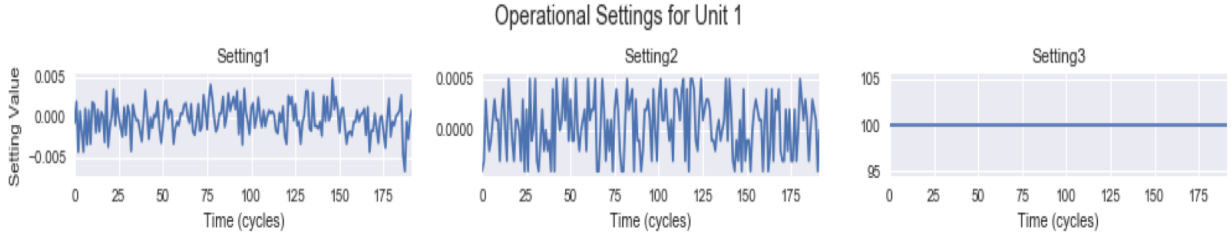


The sensor value readings over the course of the engines lifetime if what will be used to predict the engines RUL. The figure below shows the sensor readings for engine unit 1 over its lifetime. We can see that some sensors increase over time, some decrease and some don't change at all. We can also see that each sensor value contains some amount of noise.

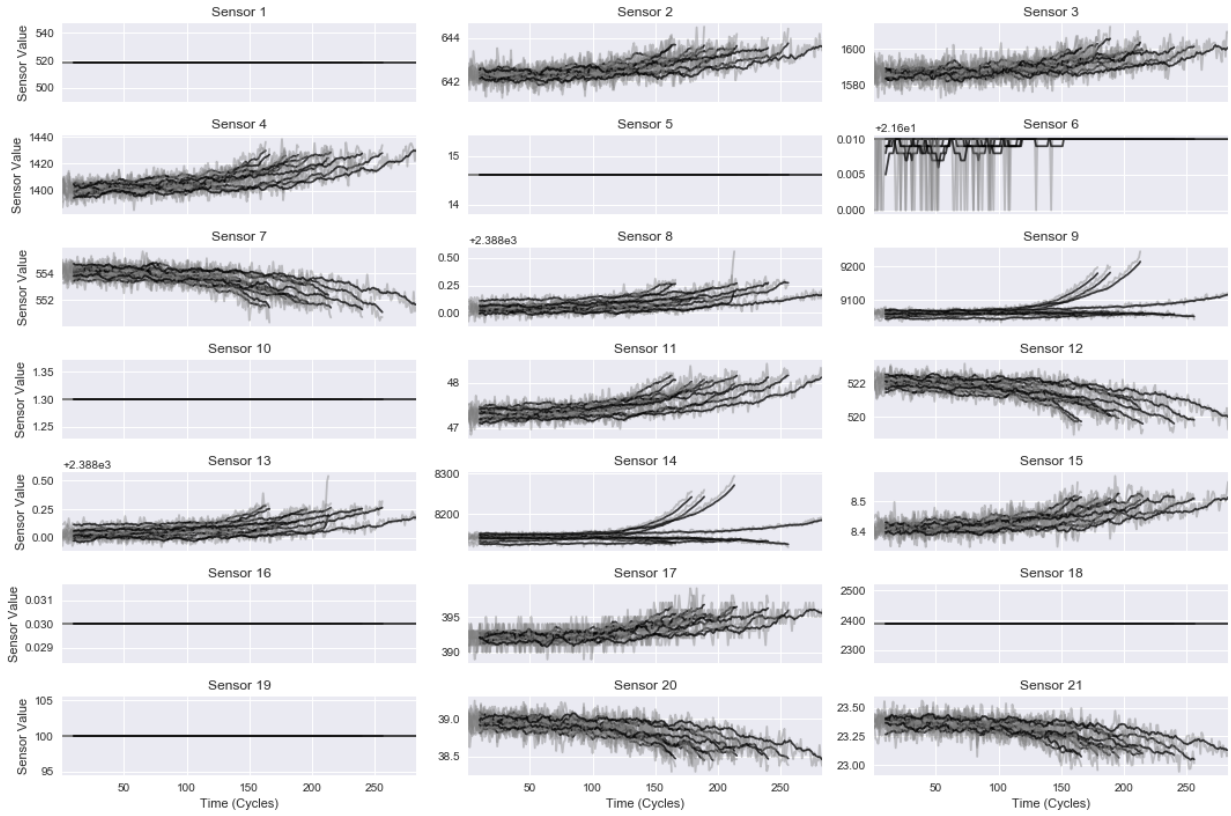
Sensor Traces for Engine Unit 1:



Each engine also operates under certain settings which change over time, as shown below.

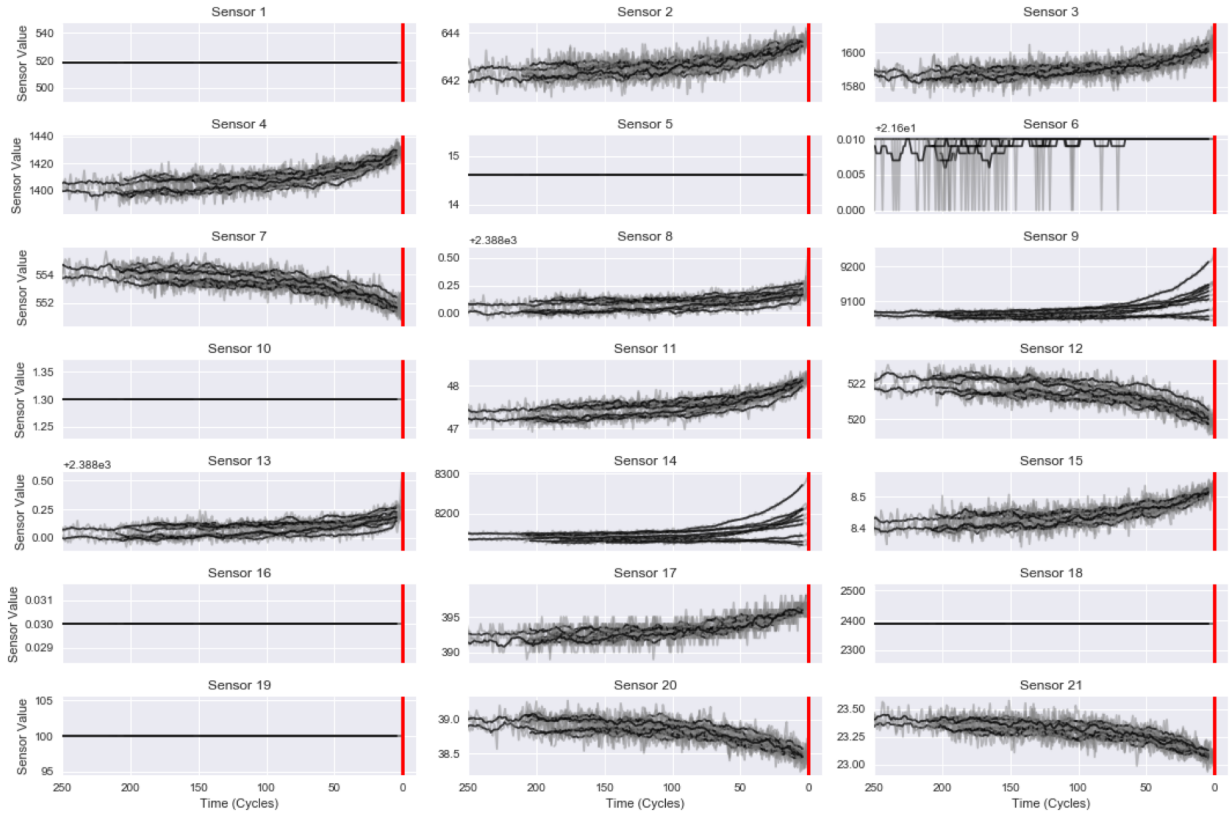


Each engine starts with a different level of wear and is run until failure. The training example contains the sensor values at each cycle up until failure. The figure below shows the sensor values from a random selection of 10 different engine units against time.

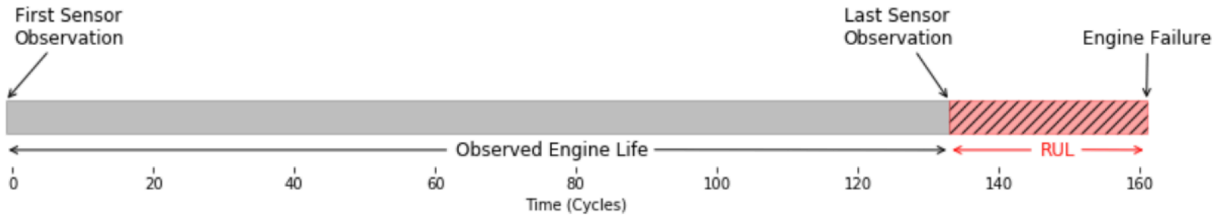


Here we can see that each engine has a different lifetime and failure pattern, which means we can't directly compare the values for the same cycle across different engine units.

Since we know the cycle that each engine unit fails on, we can compute the number of cycles until failure at each time step, which is the engines lifetime minus the the current cycle. The figure below shows the engines sensor readings plotted against its time before failure. Each engine stops at cycle 0 as indicated by the red line.

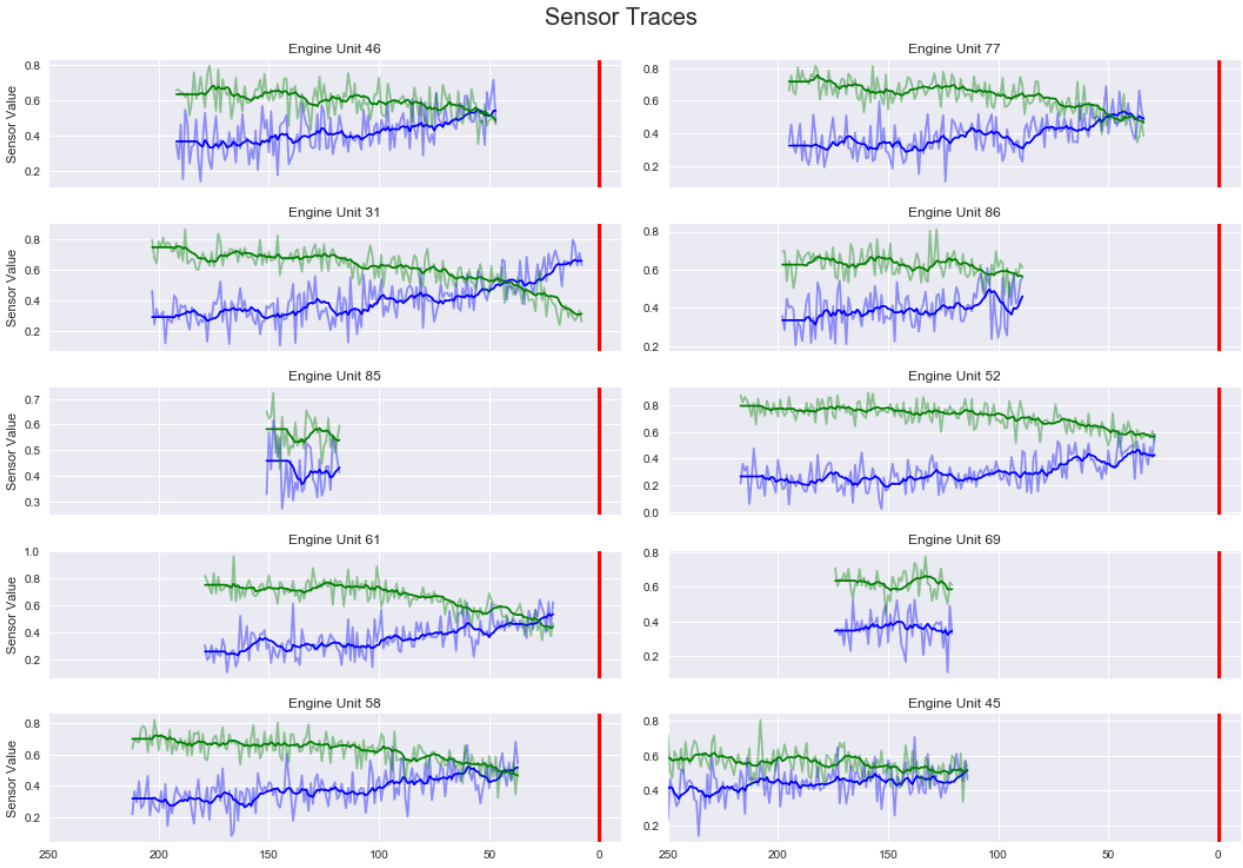


The Prediction Challenge:



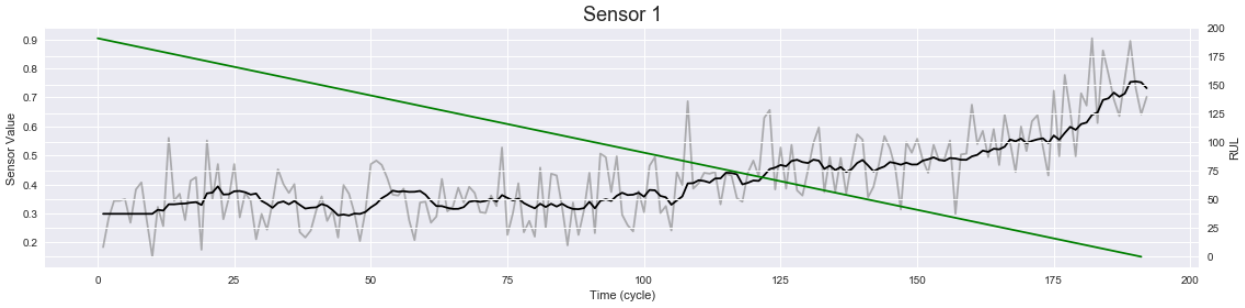
After observing an engines sensor value readings for some amount of time, the challenge is to predict the amount of time (cycles) that the engine will continue to run before it fails.

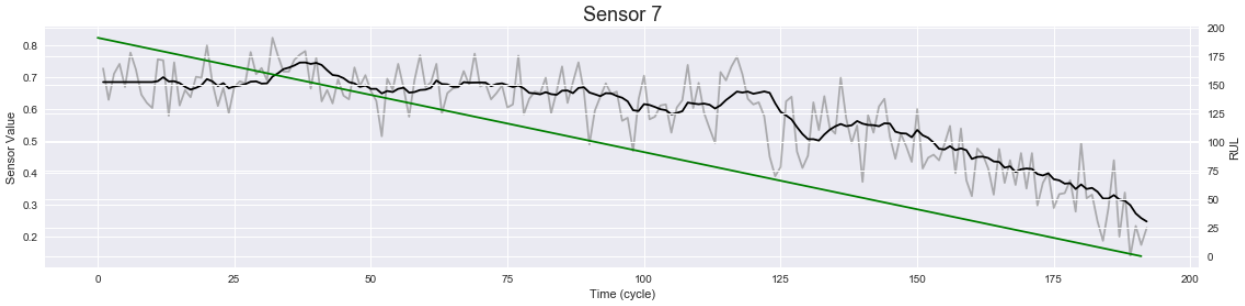
The plot below shows the available sensor value readings for sensor 2 and sensor 7 for a selection of engines in the test set against the engines remaining life, with the red line showing when the engine will fail. We can see the variance between the length of the observed sensor readings and the RUL we are trying to predict.



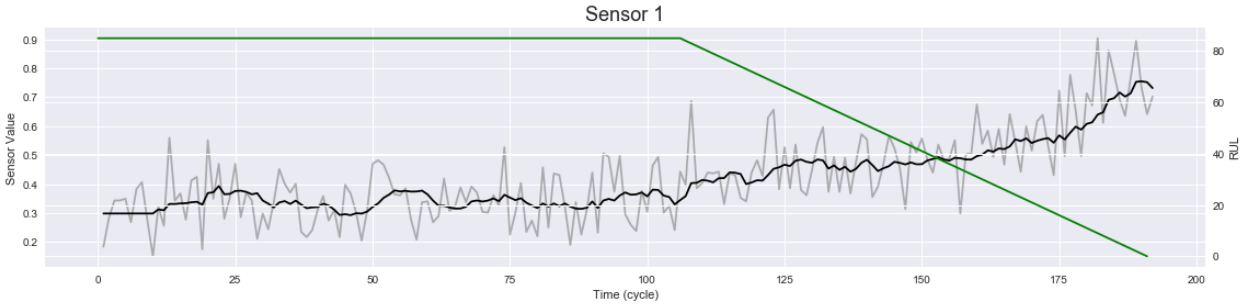
This prediction relies on there being enough information in the sensor value readings to estimate the state of the engines health. The plot below shows the distribution of sensors 2 and 7 for each engine in the training set, where healthy values in green are taken from the first 20 cycles of the engines life, and unhealthy values in red are from the last 20 cycles of the engines life.

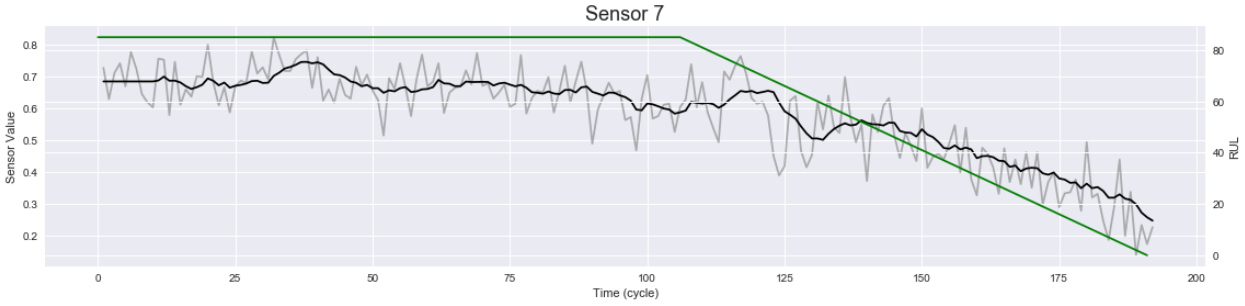
We can see that the distributions are quite different, with sensor 2 values higher for an unhealthy engine while sensor 7 values are lower.


Sensor 1
Sensor Value
RUL
Time (cycle)


Sensor 7
Sensor Value
RUL
Time (cycle)

RUL


Sensor 1
Sensor Value
RUL
Time (cycle)


Sensor 7
Sensor Value
RUL
Time (cycle)

Algorithms and Techniques

To predict the RUL based of the sensor values readings we need to find some functional relationships between the observed sensor values and the RUL of the engine. We can also include the aggregated rolling mean and standard deviation features in the function to predict the RUL value.

The function will therefor be of the form:

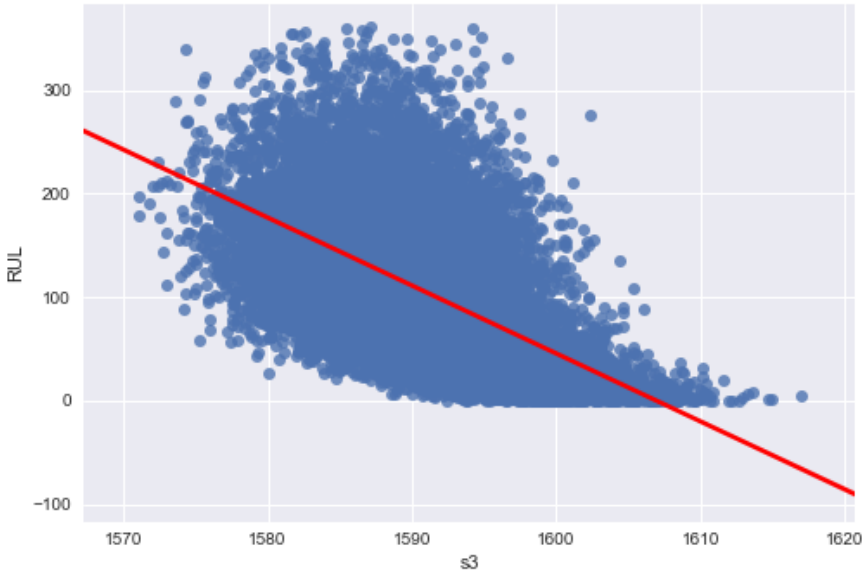
$$RUL_j(t) = f(S_{j,2}, S_{j,3}, \ldots S_{j,21}, Mean_{j,2}, Mean_{j,3}, \ldots Mean_{j,21}, Std_{j,2}, Std_{j,3}, \ldots Std_{j,21})$$

Linear Regression:

A simple first approach is to model the RUL as a linear combination of the sensor values.

$$RUL = \theta_0 + \sum \theta_1 S(t)$$

This is illustrated below, where each blue dot is a particular value of s3 against the correspond RUL for each time cycle for the engines in the training set. The red line is the line of best fit of the data.



From the plots we've seen, it would appear that the relationship between sensor values and RUL is not linear and so the Linear Regression / Logistic Regression models may be too simple to capture this relationship and give accurate predictions.

For this reason I use A Random Forrest model and a LSTM based neural network to try and capture the complexity.

Random Forrest:

Random Forrests are an ensemble method for regression and classification tasks. The main idea is that a group of weak learners (decision trees) can come together to form a strong learner (the forrest). In a decision tree, each datapoint (combination of sensor values and corresponding RUL value for each time cycle) traverses a tree of logical rules where it is bucketed into smaller and smaller sets. The tree learns what logical rules to apply to eventually produce an outcome corresponding to our target variables (RUL, label1 or label2). The rules are learned such that, at each split, the most information about the data is gained by making the split. For example, it may be the case that if s2 > X, and s7 < Y than label2 = 1.

In a Random Forrest model, a specified number of trees are trained on randomly sampled subsets of the training data. When making a prediction on a new datapoint, the datapoint is fed through each tree and an average or weighted average of the predictions from each tree is used as the actual prediction.

LSTM Neural Network:

The Random Forrest and Linear Regression models make a prediction at a specific time cycle based only on that cycles sensor values and therefor uses none of the sensor value information leading up to that time cycle.

The Long Short Term Memory (LSTM) neural network is a recurrent network that is able to use the previous sensor values to make a prediction on the current RUL. This provides a huge advantage since it is not just a functional mapping of current sensor values to RUL, but is also capable of learning patterns which occur in the sensor data and use these to make a prediction as well. Since an engine that has experienced a failure event will probably exhibit fairly consistent behaviours which are represented by the sensor readings up until it fails, this seems like a very suitable method to make these kind of predictions. For example the LSTM model is may learn that if s2 has increased at a certain rate to the current value over the past 50 cycles while s7 has decreased at a certain rate to the current value over the past 50 cycles, then the RUL = X. This also allows it to create its own features which are most suitable to the task rather than depending on feature engineering creating the rolling mean and std aggregated features, which may not be the most suitable or the only suitable features relevant to the task.

Recurrent Neural Networks (RNN's) contain a loop so that a copy of the output is fed back into to the input. This allows the decision made at time step t-1 to influence the current decision made at time step t and so recent past and present information is used to make a prediction.

The function could then be formulated as:

$$RUL = \phi(\theta_1 S_t + \theta_2 S_{t-1} + \theta_2 S_{t-2} + \ldots + \theta_n S_{t-n})$$

For some specified n.

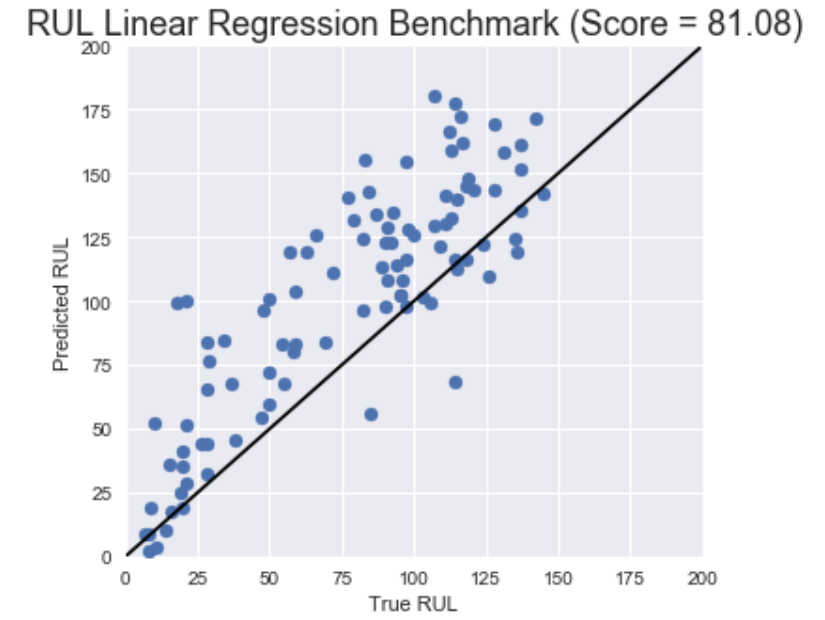
In LSTM networks each unit also includes another input which is the memory of the previous unit and so it makes decisions based on current input previous inputs and previous memory. It then generates a new output and updates its memory.

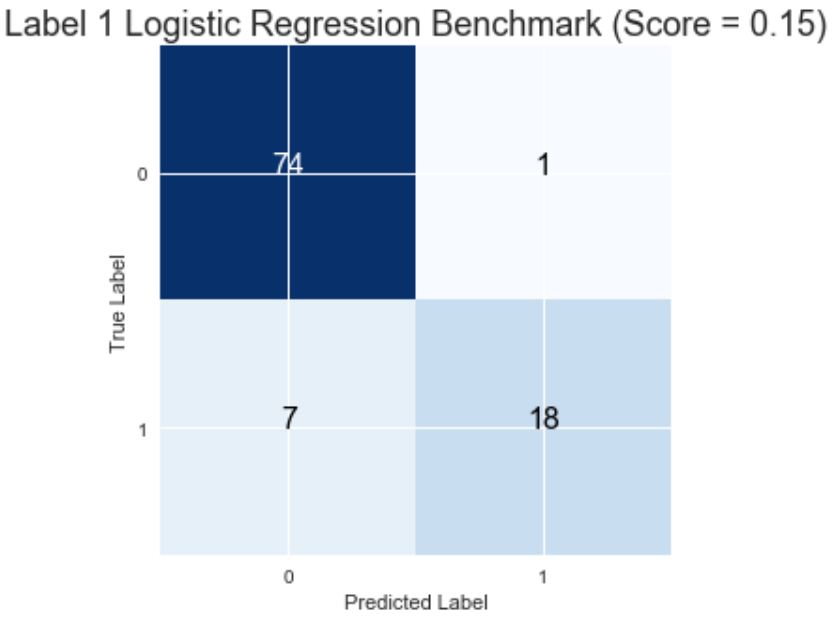
The memory cell has an input gate, output gate and forget gate which allows it to remember values over arbitrary time intervals.

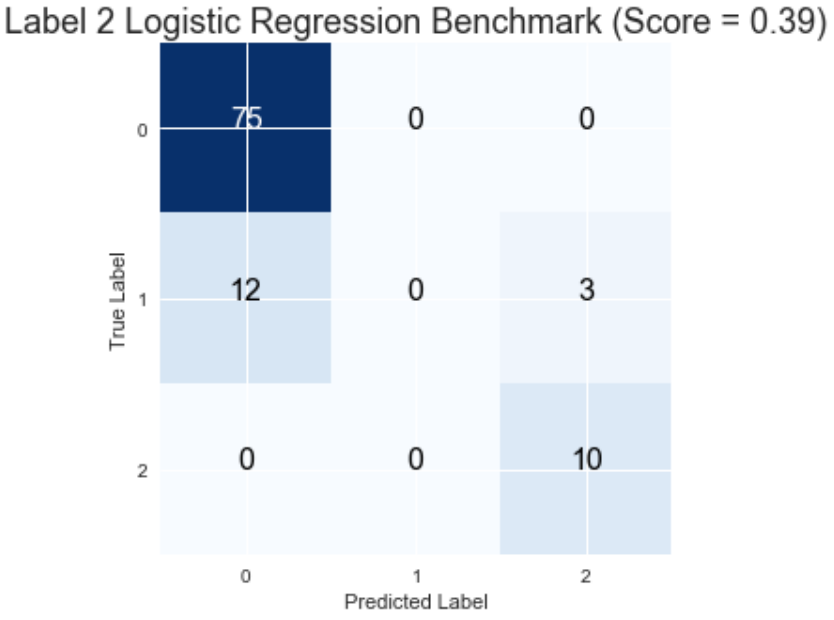
Benchmark

To gain a benchmark performance, I used a simple linear combination of the sensor values to predict the RUL. A linear regression model is used to predict the RUL, and a logistic regression model is used to predict label1 and label2 and the cost functions I defined are used to give a baseline score on the models predictions. The results are shown below.

	Benchmark_CV	Benchmark_Test
RUL	5200.74	81.08
Label1	0.06	0.15
Label2	0.18	0.39


RUL Linear Regression Benchmark (Score = 81.08)
Predicted RUL
True RUL
0
25
50
75
100
125
150
175
200


Label 1 Logistic Regression Benchmark (Score = 0.15)
74
1
7
18
True Label
Predicted Label
0
1


Label 2 Logistic Regression Benchmark (Score = 0.39)
75
0
0
12
0
3
0
0
10
True Label
Predicted Label
0
1
2

The plots above show that the simple linear model is capable of capturing some general trends in the data. We can see that the predicted RUL increases with actual RUL, and that 18 out of the 25 engines going to fail within 30 cycles were predicted, and 100% of the engines going to fail within 10 cycles were correctly predicted.

On the other hand the linear models seems to overestimate the engines RUL for almost every engine, and has a hard time differentiating engines that will fail between 15 and 30 cycles, with 12 out 15 being predicted as not failing within 30 cycles and 3 out of 15 being predicted as failing within 15 cycles.

Based on this it should be possible to get better results using a more complex model.

III. Methodology

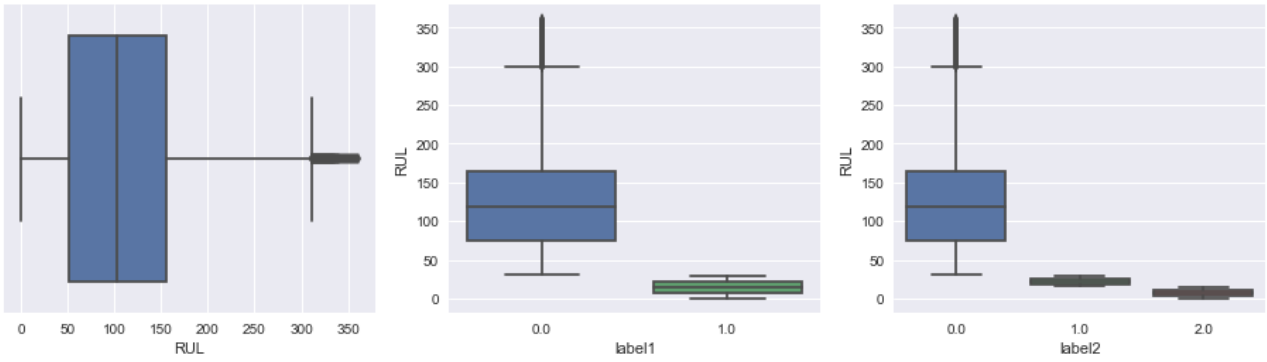
Data Pre-processing

Label Creation:

The first step is to generate the labels which are the Remaining Useful Life (RUL), label1 and label2. The RUL is calculated by taking the engines lifetime and subtracting the current cycle at each time step. Label1 is calculated by defining w1 which is the time window that we want to predict if a failure will occur, and then every column which has a RUL less than or equal to w1 is set as 1 and 0 otherwise. For label2 we define another time period w0, creating 3 time windows: RUL > w1 = 0, w1 > RUL > w0 = 1, RUL < 15 = 2.

The ground truth data is used to set the labels for the test set.

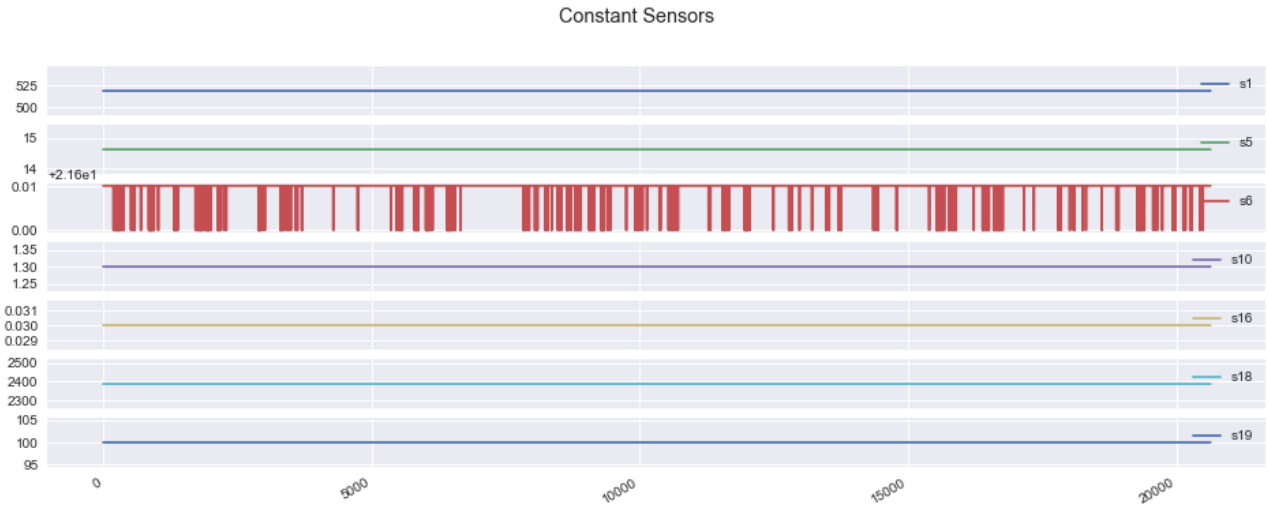
From the figure above which plots the sensor values against time for engine unit 1, we can see there are some sensor values that remain constant for the entire lifetime of the engine, namely the 7 Sensors s1, s5, s6, s10, s16, s18, and s19.

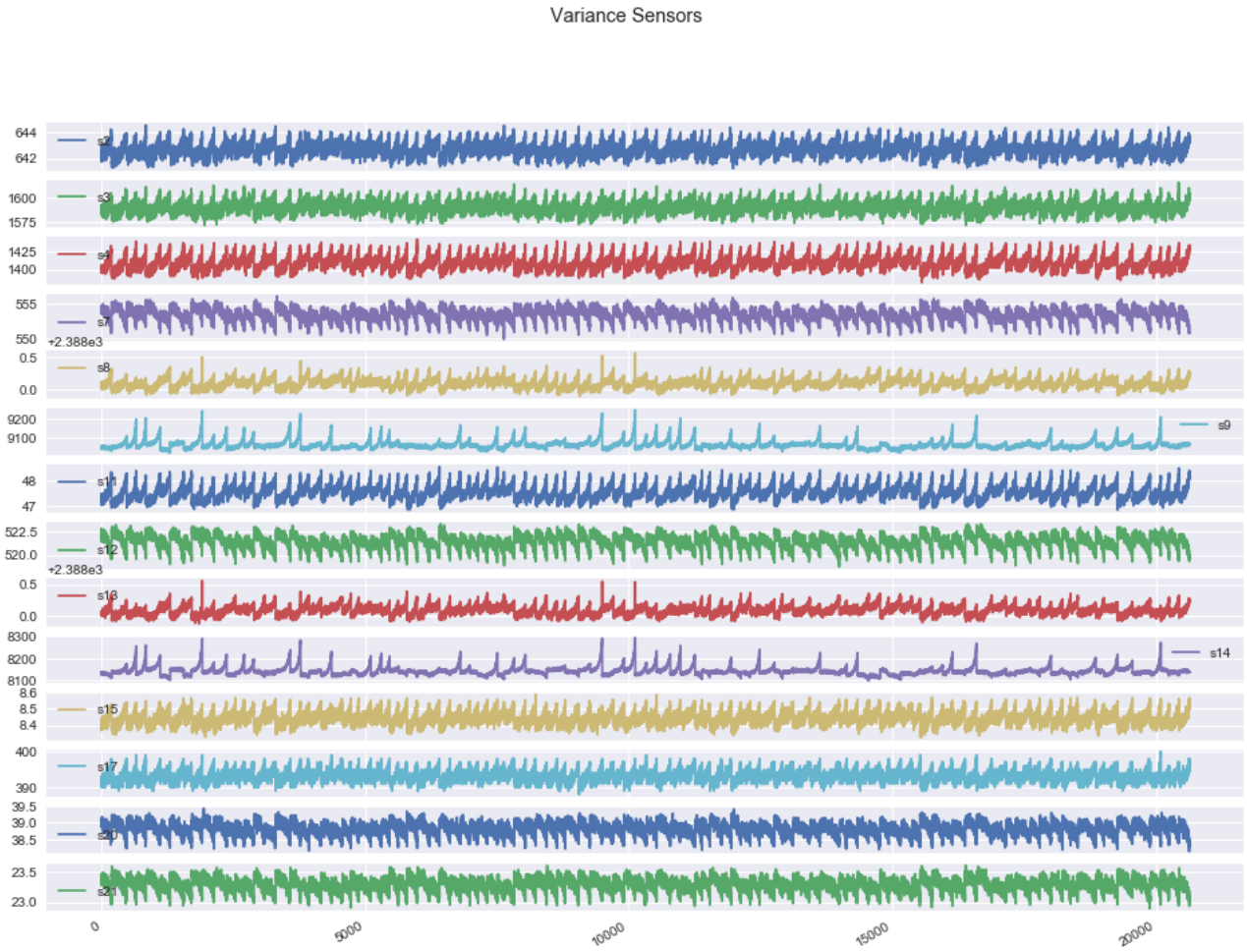


Removing Constant Sensor Values:

Subtracting the sensors min value from its max value across the all 100 engine units reveals that these sensors remain constant for all engine units in the training data with each s1,s5,s10,s16,s18 and s19 having a range of 0 and s6 having a range of 0.01. Since these sensor values remain the constant regardless of the engines health, they

provide no information that will be able distinguish between a healthy engine and an engine that is about to fail, and so they are removed from the training data, leaving 14 sensors remaining with the ranges shown below.



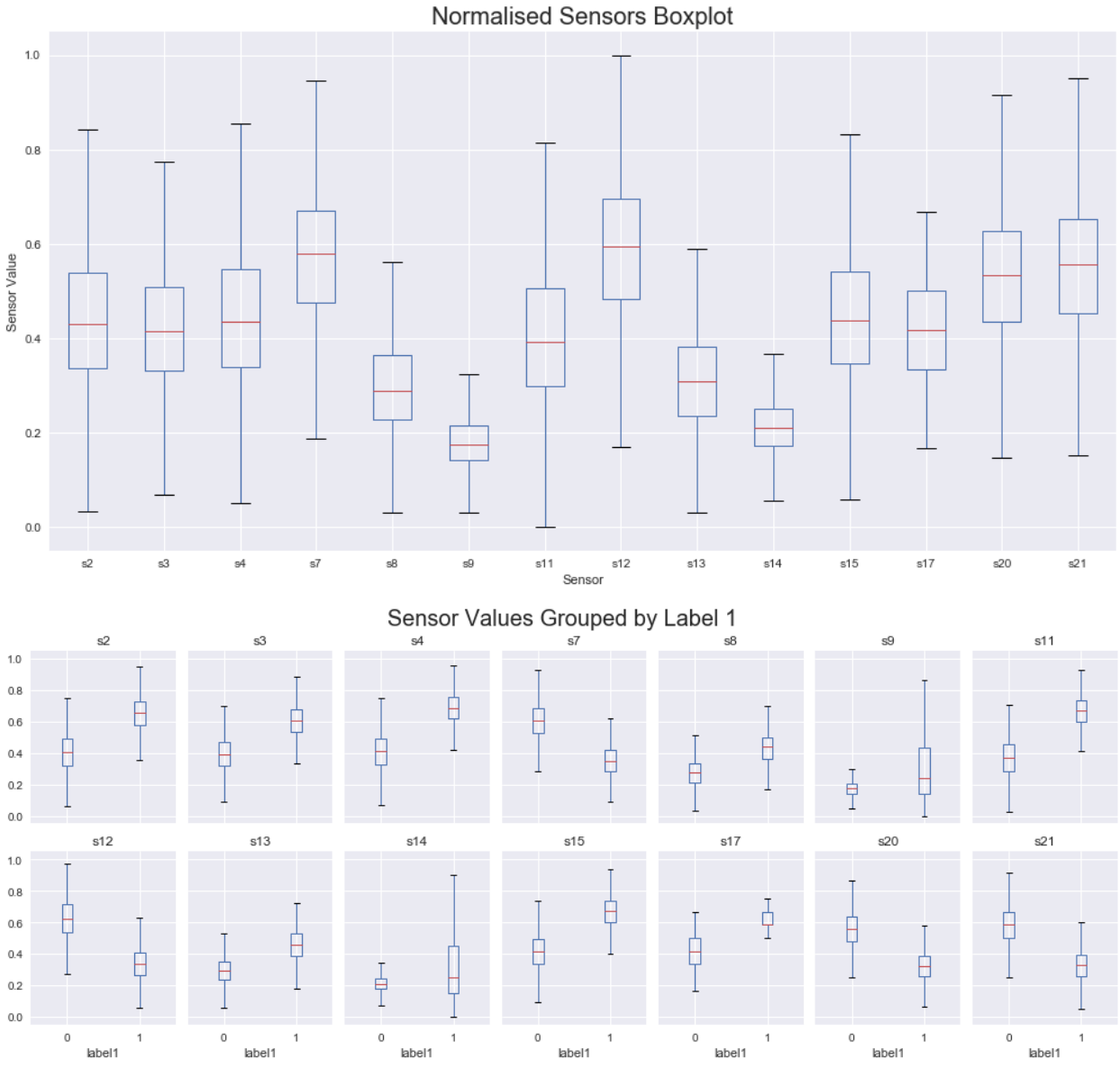


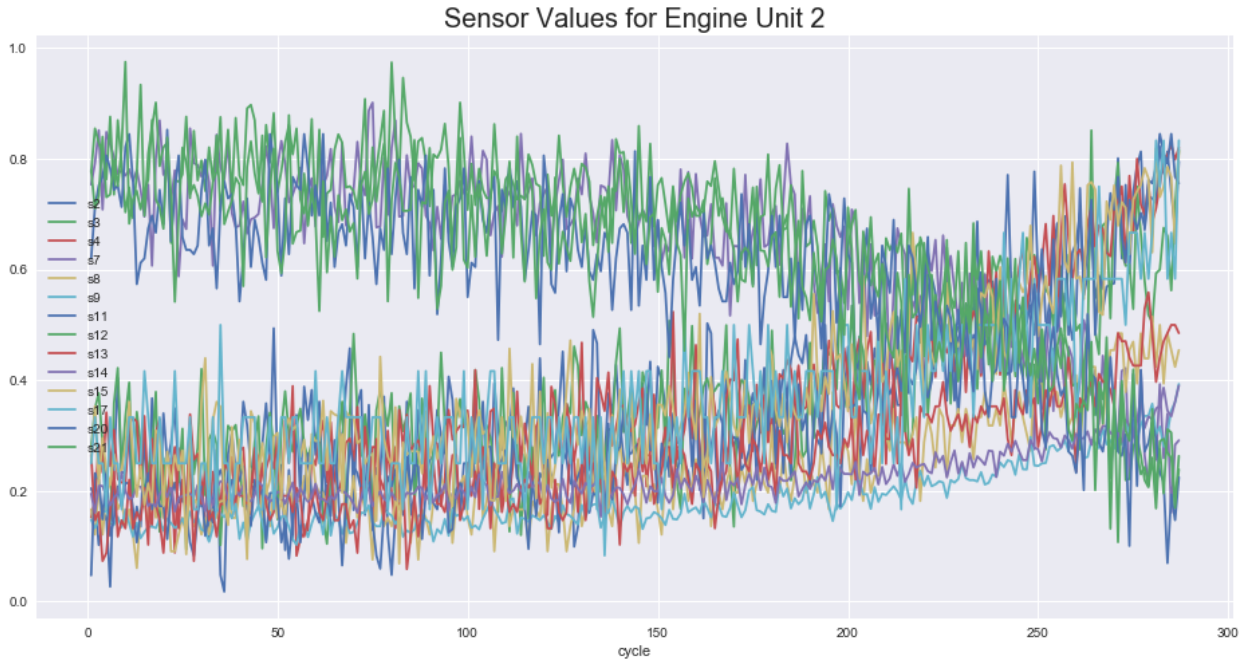
Rescale Data:

Since the sensor value readings are on vastly different scales they are rescaled into the range between 0 and 1 which allows it to be used by algorithms such as neural networks. The sklearn MinMaxScaler is used to achieve this.

A copy of the RUL value is also scaled for use with the LSTM networks which output a prediction between 0 and 1. This is then inverse scaled back to get a prediction on the original scale.

The rescaled data allows us to easier visualise the relationships between the sensor values and the target values as shown in the plots below.



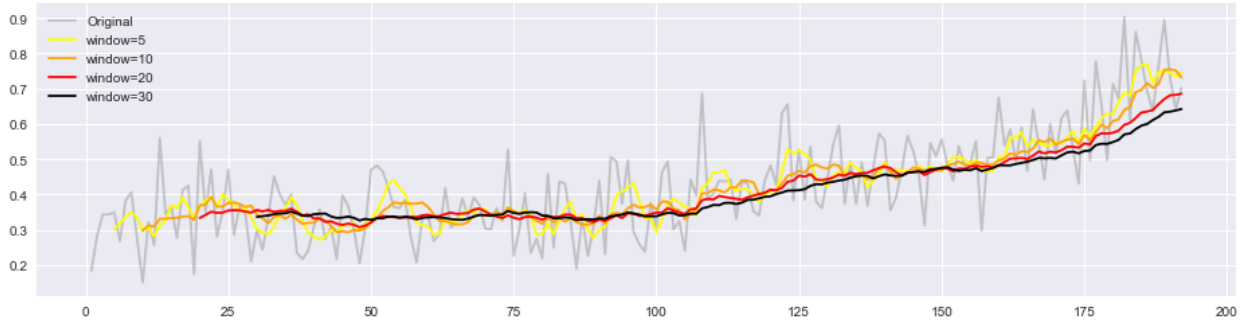


Aggregated Features:

In the above plot of sensor 2 (blue) and sensor 7 (green) against time, we can see that the sensor values contain a lot of noise. This is shown by the value jumping up and down between each cycle. These constant upward and downward jumps don't give

much information about the health of the engine since they appear to happen pretty consistently throughout the life of the engine. We can see, however that the general trend of the sensor values is to rise for sensor 2 and fall for sensor 7 towards the end of the engines life, presumably after a failure event has occurred around the 110th cycle which causes the sensors to exhibit these changes.

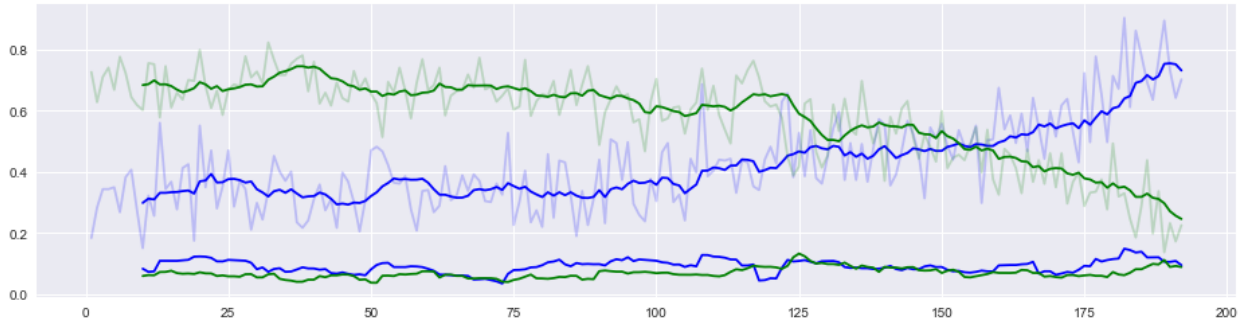
To capture the general trend of the sensor values we can use a rolling window over the data with various aggregations such as the mean and standard deviation.



The plot above shows the rolling mean for windows of 5, 10, 20 and 30 cycles. We can see that increasing the window size increases the smoothing of the data.

In this experiment I add a mean and std feature for each sensor value using a window of size 10 using the pandas rolling function.

The plot below shows the mean (top) and std (bottom) features.



Implementation

The train and test datasets are first preprocessed by creating the RUL, label1 and label2 target variables, then undergoing scaling to put the sensor features and RUL on a scale between 0 and 1 and then rolling mean and std features using a window of 10 are added.

Then the train and test sets are made from the data. This is straight forward for the Linear Regression and Random Forrest test sets which only require taking the row corresponding to the last observed time cycle for each engine id to make up the test set.

The LSTM dataset is more complicated however, since we need to put it into sequences with the input a 3D matrix of shape (batch_size, sequence_length, number_of_features), for example with the defined sequence length of 60 using the 14 sensor features, the input is of shape (14631, 60, 14), meaning we have made 14631 sequences of length 60 from our training set.

The following models are then used:

RUL: Linear Regression, Random Forrest Regressor, LSTM network with linear output activation

Label1: Logistic Regression, Random Forrest Classifier, LSTM network with sigmoid output activation layer

Label2: Logistic Regression, Random Forrest Classifier, LSTM network with softmax output activation layer

For each model, the training set is split up into 10 groups made up of 10 engine units each. On the linear and random forrest models 10 fold cross validation is used on the training data, while on the LSTM models, 3 groups are randomly picked out of the possible 10 to be used as the validation sets due to the long training times and therefor cross validation is run 3 times using 90 engines to train the model and 10 engines to validate each time. An average of the cross val scores is given as a cv score. For each model.

Model hyper-parameters are tuned using cross validation and then the resulting model is fit to the entire training set and used to make predictions on the test set which are scored using the metrics as defined above.

The resulting LSTM model for RUL predictions is shown below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_308 (LSTM)              (None, 60, 50)            13000
_________________________________________________________________
dropout_304 (Dropout)        (None, 60, 50)            0
_________________________________________________________________
lstm_309 (LSTM)              (None, 20)                5680
_________________________________________________________________
dropout_305 (Dropout)        (None, 20)                0
_________________________________________________________________
dense_145 (Dense)            (None, 1)                 21
_________________________________________________________________
activation_145 (Activation)  (None, 1)                 0
=================================================================
Total params: 18,701.0
Trainable params: 18,701.0
Non-trainable params: 0.0
_________________________________________________________________
None
```

The RUL prediction network uses a fully connected Dense layer made of 1 unit with a linear activation. Since the RUL target value was normalised, the predicted value will be a value between 0 and 1. The predicted value in then inverse transformed back to the original range.

The label1 prediction network has a Dense layer with a sigmoid activation, with the prediction giving the probability of the output belonging in class 1. This output is rounded to the nearest value (0 or 1) to give the predicted class.

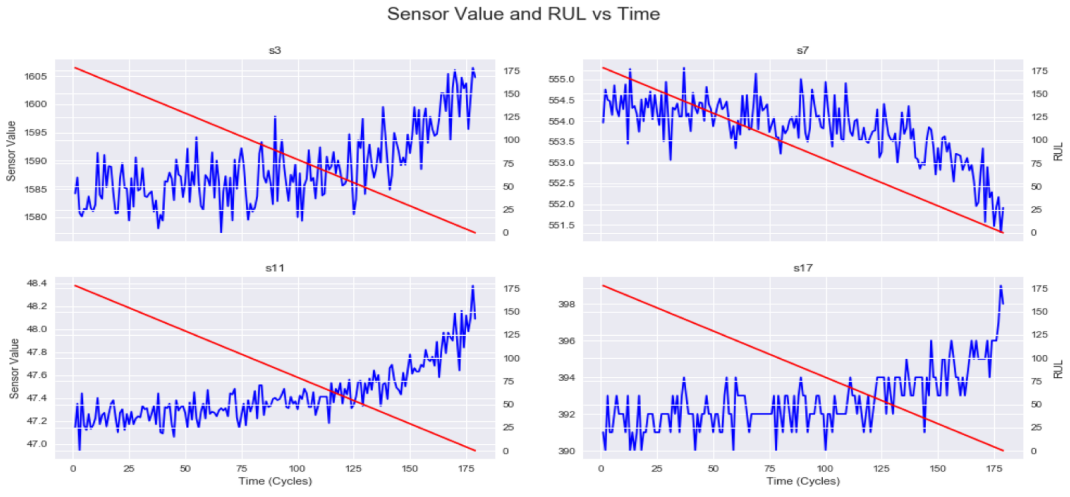
The label 2 prediction network has a Dense layer made up of 3 units with softmax activation. This gives the probability distribution of belonging to each of the 3 classes. The output with the highest probability is chosen as the predicted class.

The figure below shows the performance of a Linear Regression model using all sensor values as features on the training data. Each point represents a single prediction made by the model plotted against its actual RUL value. The black line represents a perfect prediction (prediction == actual).

Using a max RUL value:

Taking a step back, I considered the fact that the engines in the training data experience a failure event which occurs at some point into the engines life cycle, at which point the sensor values start changing in reaction to the degradation health of the engine up until its failure where the RUL is 0, while the RUL value linearly decreases from the engines first cycle until its last.

This means that, for the early cycles of each engine, we are training the models to predict a constantly decreasing RUL value based of fairly consistent sensor data, as show in the plot below.

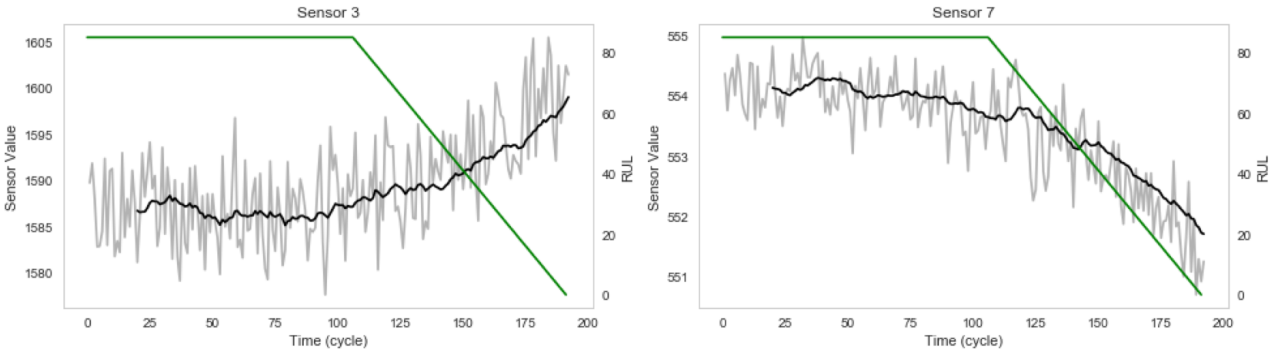


If we think of the RUL as an indicator of the engines health, then it should remain constant while the engine is operating under normal conditions. When the engine experiences a failure condition at a set time before its end of life, then the engines

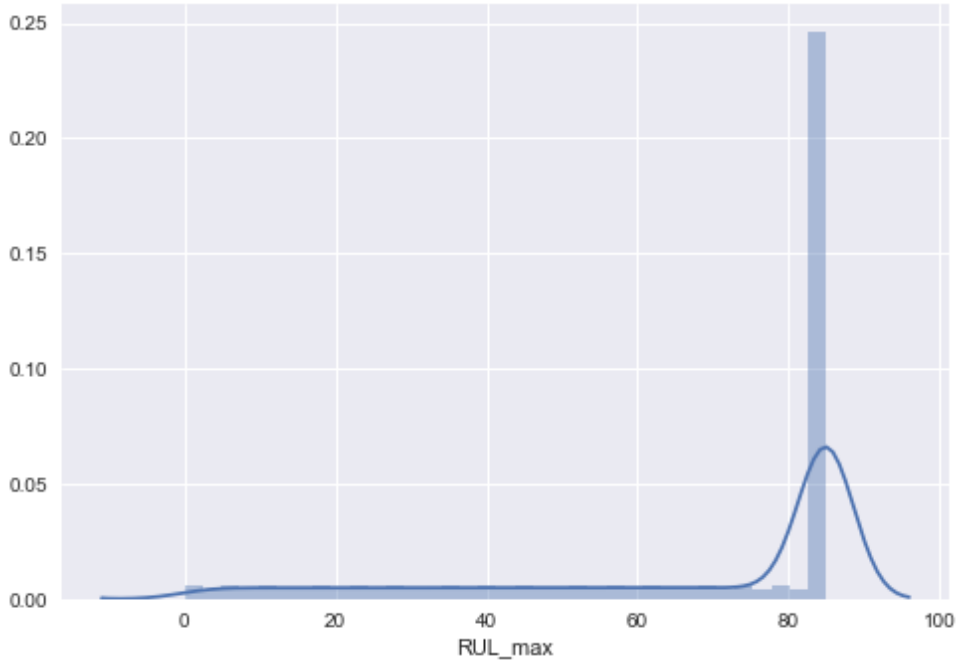
health deteriorates which is reflected in the sensor values and then the RUL should decrease to reflect the deteriorating health.

Training the models based on the engines early cycles when the sensor data is fairly constant is likely contributing to training error, and minimising these cycles will have no effect on the ability to predict the RUL when an error occurs.

The training data was modified to have a maximum RUL value of 85 giving a constant RUL value of 85 before it linearly decreases down to 0. This is to reflect that an engine should have a constant RUL life while in healthy operating conditions up until an event occurs which causes the engines health to degrade until it fails.



The distribution of the RUL with max value of 85 value is shown below.



Refinement

GridSearchCV was used to refine the Random Forrest model using the following parameters grids for RUL and label1 , label 2 respectively:

```
rul_param_grid = {"n_estimators" : [5, 10, 50],
                  "max_depth": [None, 5, 20],
                  "min_samples_split": [2, 5],
                  "min_samples_leaf": [1, 5],
                  "max_features" : ["auto", "log2"]
                 }

label_param_grid = {"n_estimators" : [5, 10, 50],
                    "max_depth": [None, 5, 20],
                    "min_samples_split": [2, 5],
                    "min_samples_leaf": [1, 5],
                    "criterion" : ['gini', 'entropy'],
                    "max_features" : ['auto', 'log2']
                   }
```

The best parameters were:

RUL:

```
{'max_depth': None,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 5}
```

Label1:

```
{'criterion': 'entropy',
 'max_depth': 20,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 5}
```

Label2:

```
{'criterion': 'entropy',
 'max_depth': 20,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 5}
```

For the LSTM network various combinations of parameters were tried and validated on the 3 validation groups of 10 engines each to give a mean score.

This included adjusting the number of layers (2 and 3), the number of units in each layer (100-50, 50-20, 50-20-5, 10-5) the loopback sequence length of the data(40, 50, 60), the dropout probabilities (0.2 to 0.5) and the batch size (100, 200, 600) which gave various results ranging from 44 to 1000.

The best model gave a mean CV score of 45 with the rolling parameters:

Lookback: 60

1st layer with 50 LSTM units

2nd layer with 20 LSTM units

Dropout of 0.5 after each LSTM Layer

Training Batch Size of 200

Prediction Batch Size of 1

IV. Results

Model Evaluation and Validation

The results of the original models are summarised below.

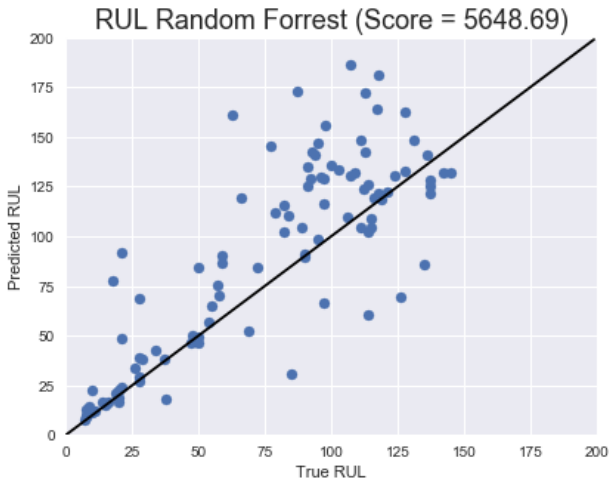
Random Forrest Models:

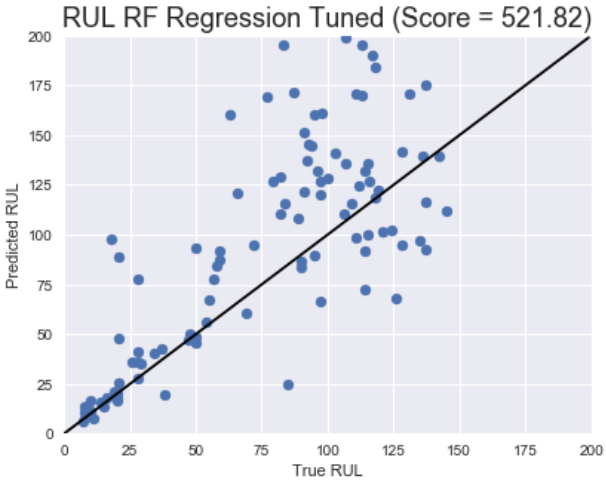
Before Tuning:

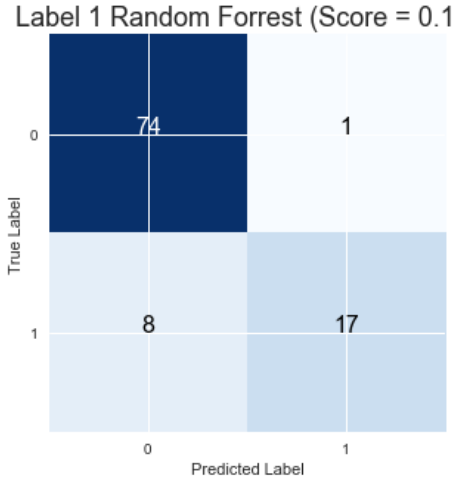
	RandomForrest_CV	RandomForrest_Test
RUL	21410.33	5648.69
Label1	0.07	0.17
Label2	0.15	0.32

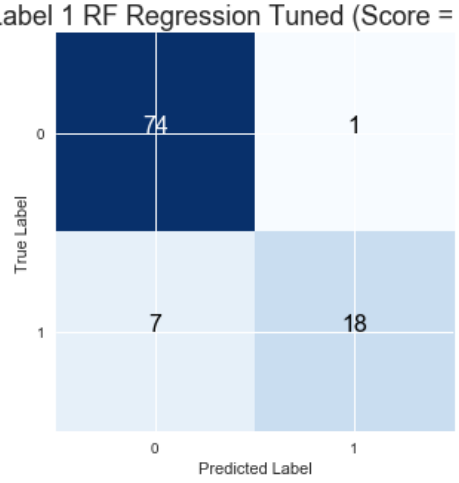
After Tuning:

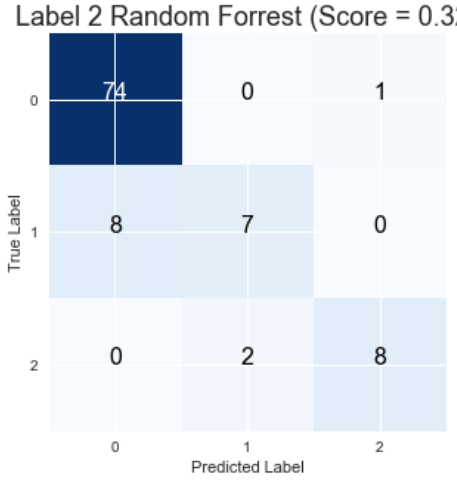
	RandomForrest_CV_tuned	RandomForrest_Test_tuned
RUL	46637.68	521.82
Label1	0.07	0.15
Label2	0.16	0.38

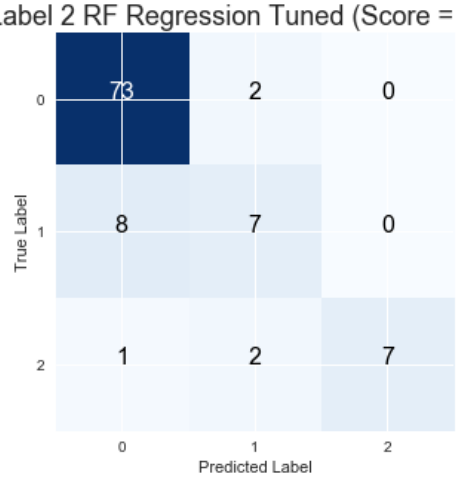

RUL Random Forrest (Score = 5648.69)
Predicted RUL
True RUL
0
25
50
75
100
125
150
175
200


RUL RF Regression Tuned (Score = 521.82)
Predicted RUL
True RUL
0
25
50
75
100
125
150
175
200


Label 1 Random Forrest (Score = 0.17)
True Label
Predicted Label
74
1
8
17
0
1

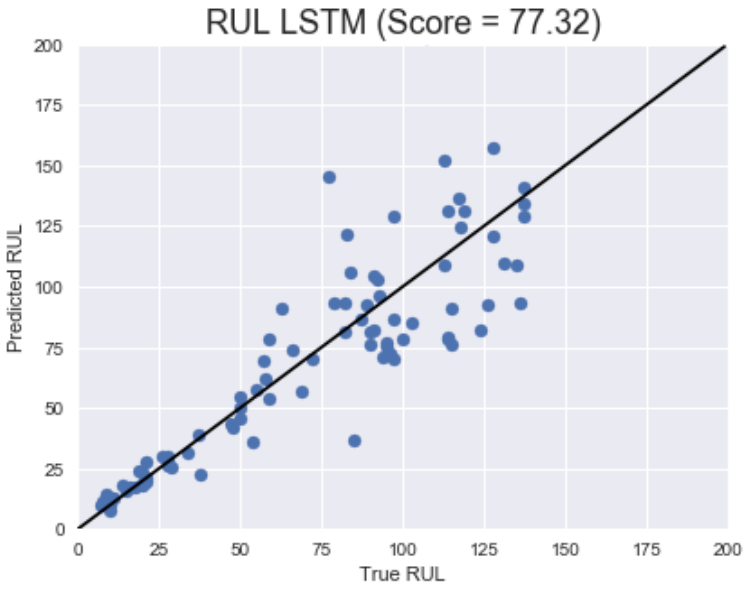

Label 1 RF Regression Tuned (Score = 0.15)
True Label
Predicted Label
74
1
7
18
0
1

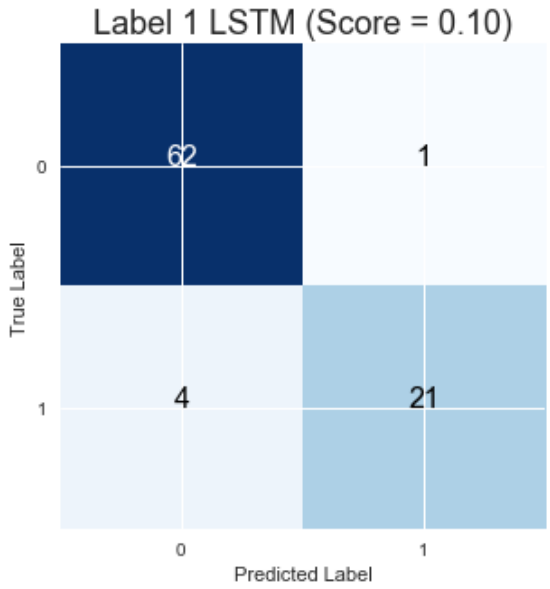

Label 2 Random Forrest (Score = 0.32)
True Label
Predicted Label
74
0
1
8
7
0
0
2
8
0
1
2

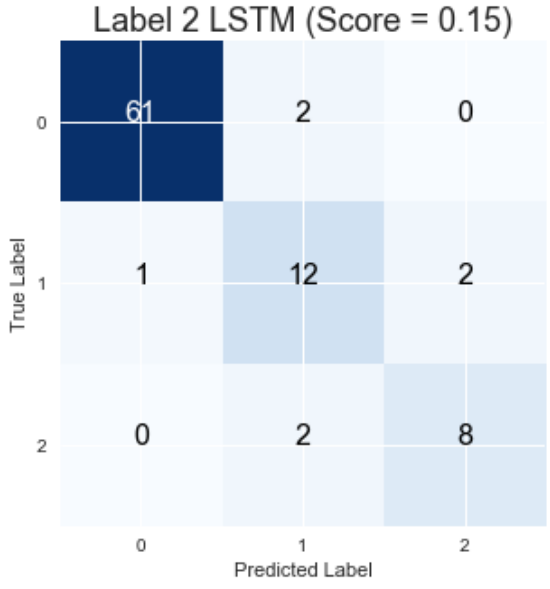

Label 2 RF Regression Tuned (Score = 0.38)
True Label
Predicted Label
73
2
0
8
7
0
1
2
7
0
1
2

LSTM Models:

	LSTM_CV	LSTM_Test
RUL	64.91	77.32
Label1	0.05	0.10
Label2	0.10	0.15


RUL LSTM (Score = 77.32)
Predicted RUL
True RUL
0
25
50
75
100
125
150
175
200

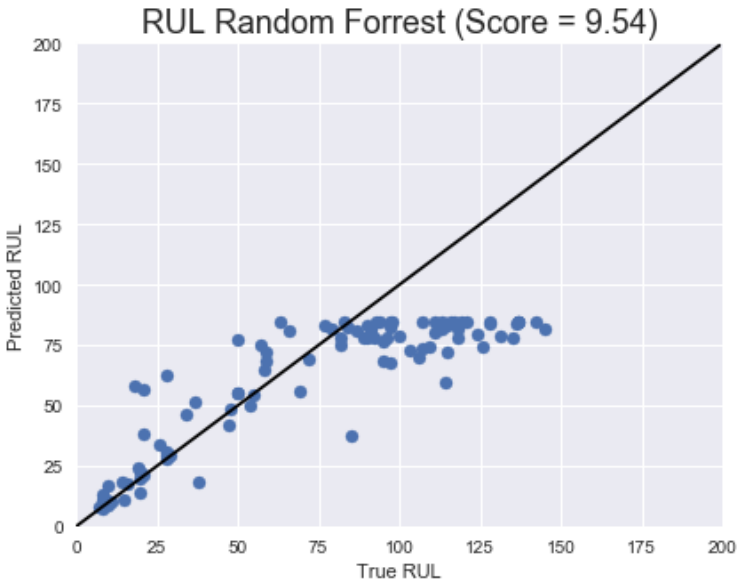

Label 1 LSTM (Score = 0.10)
True Label
Predicted Label
62
1
4
21
0
1

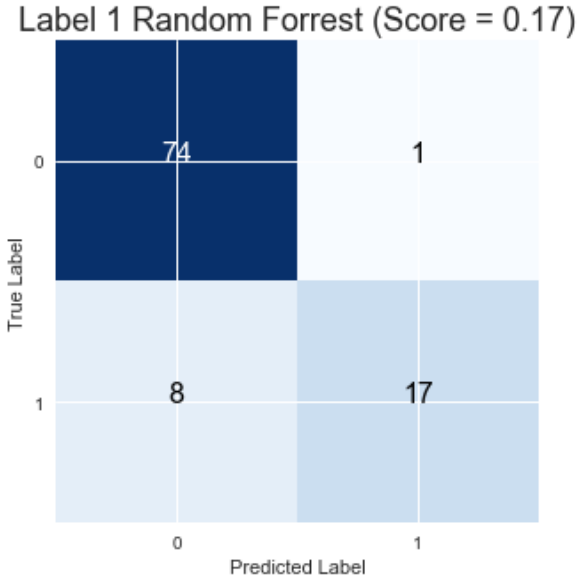

Label 2 LSTM (Score = 0.15)
True Label
Predicted Label
61
2
0
1
12
2
0
2
8
0
1
2

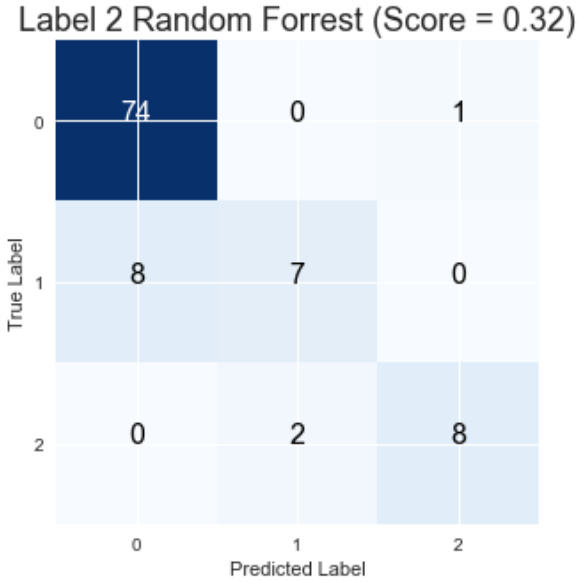
Using MAX RUL Value:

Random Forrest:

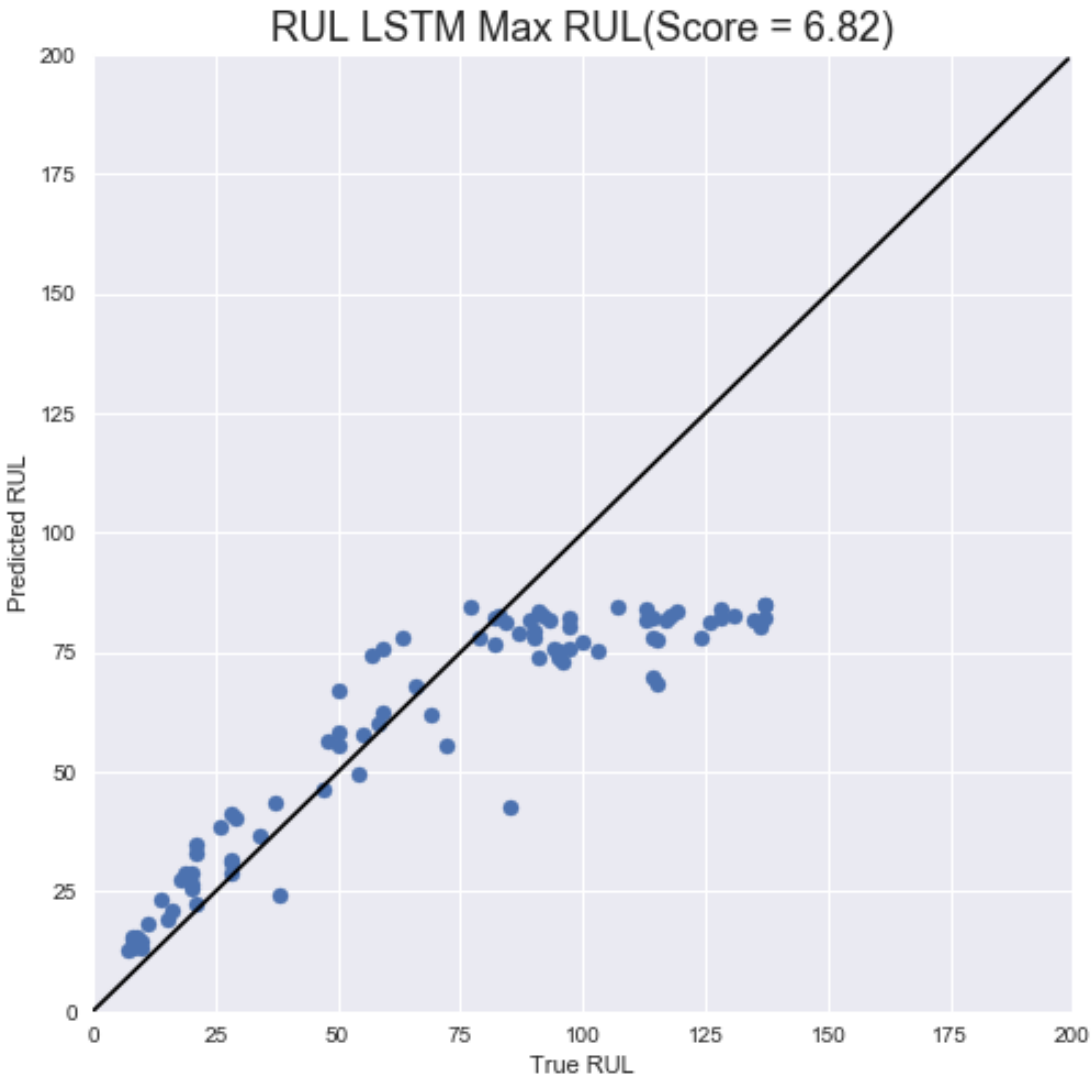
	RandomForrest_CV_RUL_Max	RandomForrest_Test_RUL_Max
RUL	1.65	9.54
Label1	0.07	0.17
Label2	0.15	0.32

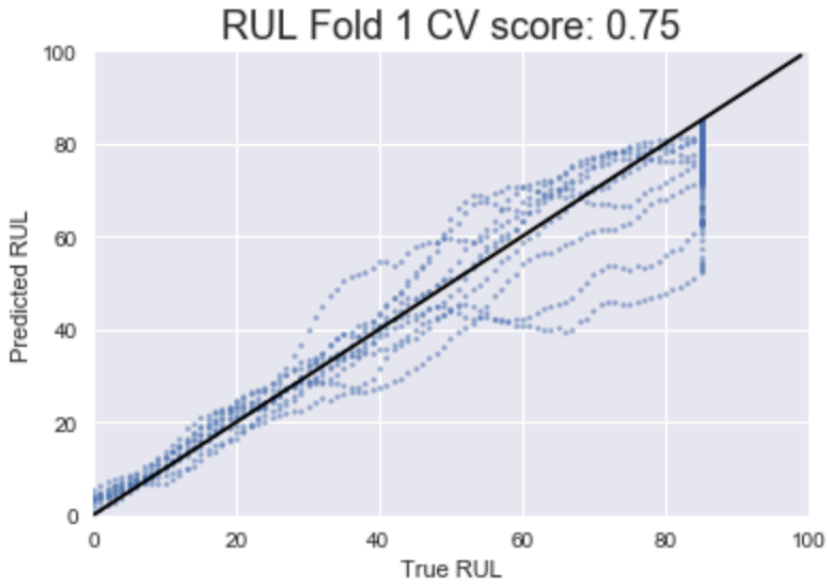

RUL Random Forrest (Score = 9.54)
Predicted RUL
True RUL
0
25
50
75
100
125
150
175
200

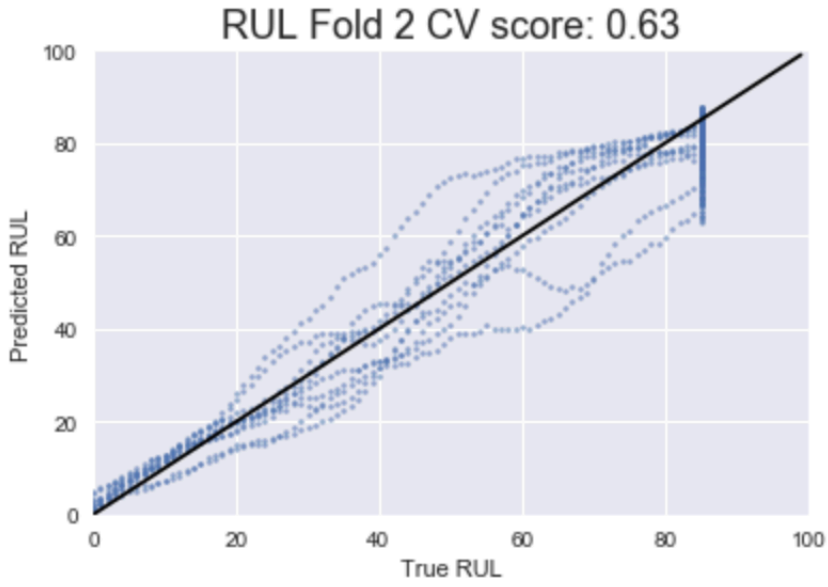

Label 1 Random Forrest (Score = 0.17)
True Label
Predicted Label
74
1
8
17
0
1

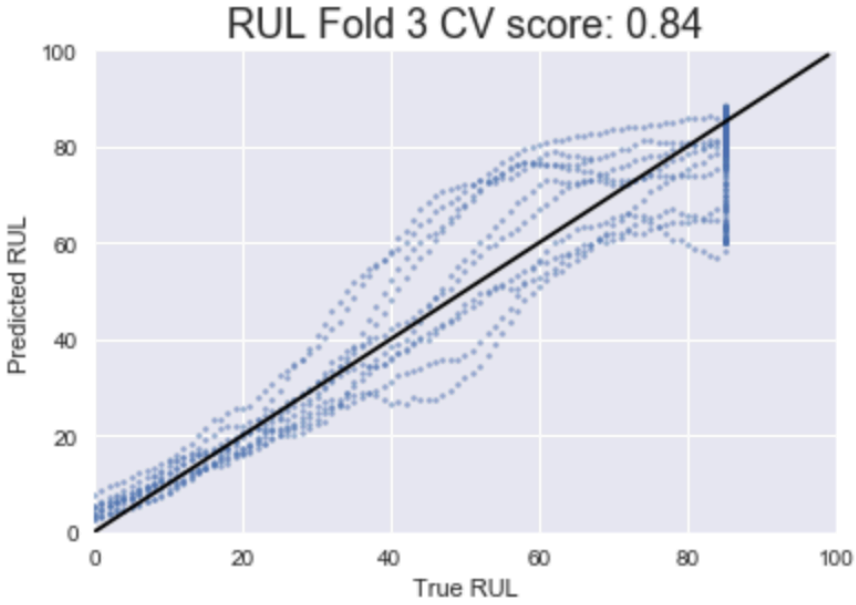

Label 2 Random Forrest (Score = 0.32)
True Label
Predicted Label
74
0
1
8
7
0
0
2
8
0
1
2

LSTM:


RUL LSTM Max RUL(Score = 6.82)
Predicted RUL
True RUL
0
25
50
75
100
125
150
175
200


RUL Fold 1 CV score: 0.75
Predicted RUL
True RUL
0
20
40
60
80
100


RUL Fold 2 CV score: 0.63
Predicted RUL
True RUL
0
20
40
60
80
100


RUL Fold 3 CV score: 0.84
Predicted RUL
True RUL
0
20
40
60
80
100

These results show a decent improvement in RUL predictions score, mainly as a result of minimising the over predictions for the RUL values at the higher end, since these are heavily penalised by the cost function.

Results Summary:

	RUL	Label1	Label2
RandomForrest_CV_tuned	46637.684844	0.072706	0.159760
RandomForrest_CV	21410.325865	0.069459	0.152878
Benchmark_CV	5200.737650	0.062334	0.180214
LSTM_CV	64.911125	0.047298	0.101454
RandomForrest_CV_RUL_Max	1.651039	0.069459	0.152878
LSTM_CV_RUL_Max	0.741079	0.047298	0.101454

	RUL	Label1	Label2
RandomForrest_Test	5648.685177	0.170000	0.320000
RandomForrest_Test_tuned	521.823052	0.150000	0.380000
Benchmark_Test	81.075776	0.150000	0.390000
LSTM_Test	77.321907	0.102273	0.147727
RandomForrest_Test_RUL_Max	9.542447	0.170000	0.320000
LSTM_Test_RUL_Max	6.820150	0.102273	0.147727

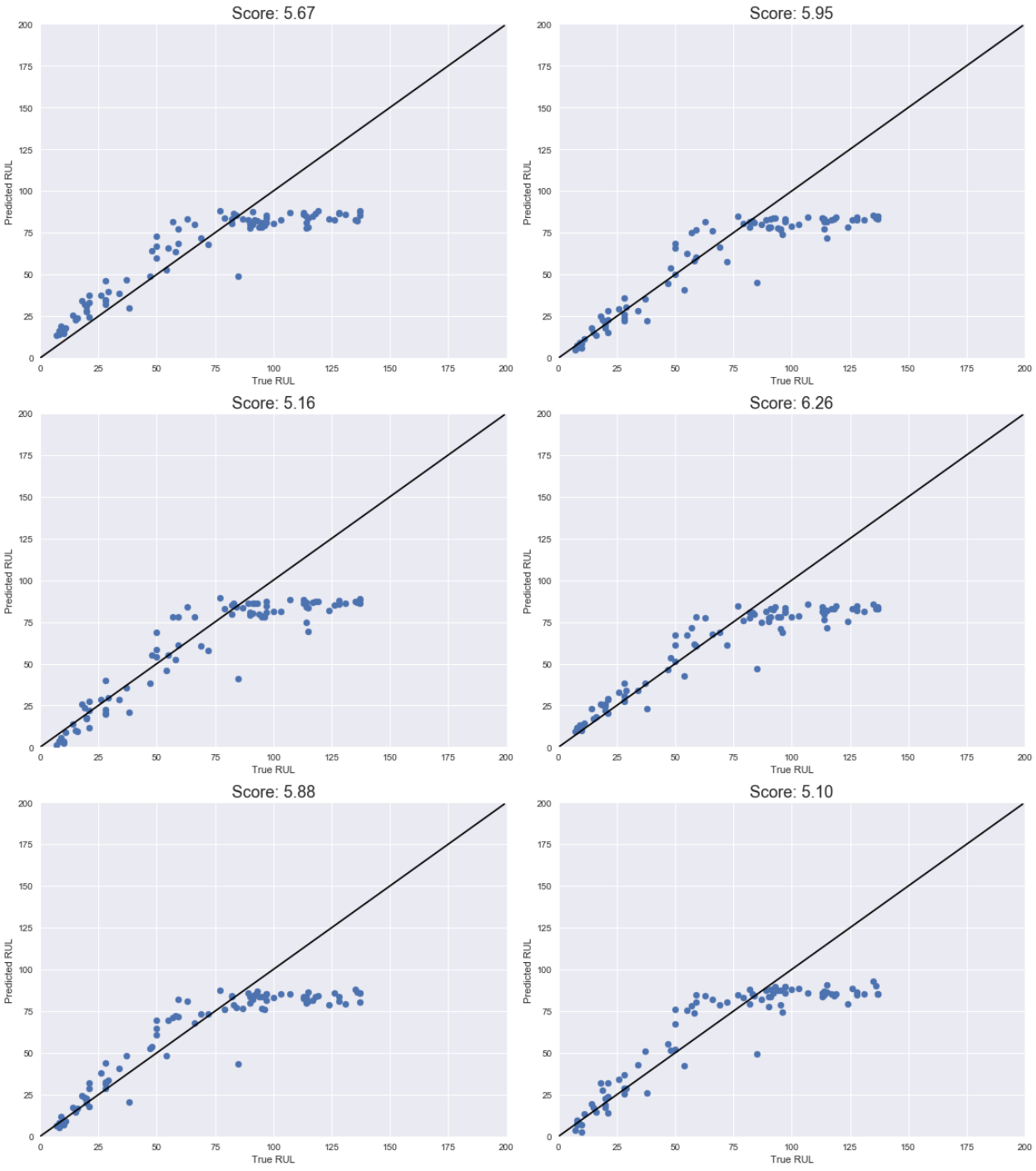
The LSTM model shows promising results, especially with the Max RUL value of 85 applied for both the Cross Validation scores on the training data and on the test data set.

When compared to the Benchmark model, the LSTM model does a much job at predicting the RUL with a score of 6.82 compared to 81 on the test data and much better cross validation scores. The LSTM model was the only model that was able to accurately predict engines that would fail between 15 and 30 cycles in the label 2

predictions, with 12 out of 15 accurately predicted where the benchmark model predicted 0 and Random Forrest predicted 7 out of 15. Only 4 out of 25 engines that would fail within 30 cycles were incorrectly predicted with the LSTM model compared to 7 for both the Benchmark and Random Forrest models.

LSTM Model Validation:

To validate the performance of the LSTM model, 6 new test sets were created using different offsets from of each engine the the last given value, i.e. the 2nd, 3rd, 4th, 5th, 6th, 7th and 8th last values for each engine. The results are shown below.



Justification

The LSTM network provides promising results for predicting the RUL, predicting if a failure will occur within 30 cycles and predicting if a failure will occur 15 cycles, between 15 and 30 cycles or not within 30 cycles. The RUL predictions can be further improved by setting a max RUL value which is meant to reflect when the engine is operating normally, then linearly decreasing when an error event occurs.

If a failure is predicted then the solution could be used to alert staff who could do further investigation into the engines health. The danger would be when the engine is about to fail and the model doesn't predict it which could lead to large costs or even catastrophic events if people rely on the predictions too much.

Therefor I think the solution would be great if it was used to give early warning alarms about potential failure conditions, or the predictions could be used as a sort of ranking system which would aid in prioritising maintenance activities. However I would be hesitant to rely upon it to be certain that the engine will not fail without other methods of confirming this.

V. Conclusion

Reflection

The first step in the project was determining what the labels we wanted to predict were and what type of machine learning algorithms would be able to solve them. The Remaining Useful Life (RUL) was calculated based on the current cycle and the cycle at which the engine failed and the label 1 and label 2 classified the RUL into specific time windows.

The next step was understanding the input sensor data and determining whether it was sufficient enough information to predict the Remaining Useful Life of the engine. From the visualisations we could see that each sensor does follow a specific degradation pattern when the engine experiences a failure event and by analysing the sensor value distribution when an engine is healthy compared to when it is about to fail we could see that the sensor values could distinguish between a healthy and unhealthy engine.

Next was determining the cost function that was used to score the models performance, or in other words to penalise incorrect predictions. The cost function was based on the cost that an incorrect prediction would cause to a company using the models in a production environment. Asymmetric cost functions were used to capture the higher associated cost of failing to predict an engine approaching failure as opposed to incorrectly predicting the engine was approaching failure when it was actually OK.

The next step was looking at some feature engineering, with the sensor values that remained constant discarded and a rolling mean and rolling standard deviation over the remaining sensor values used.

Simple linear models were used on all sensor features to determine a benchmark performance for each prediction category.

Then the data sets were split into appropriate train and test sets, with sequences of data of length 50 used for the LSTM network. The models were then trained and validated on the training data using a 5 fold group split with groups based on the engines id, then predictions were made on the test data, and the actual test values and the predictions were used to score the model.

Further exploration into the data highlighted that for much of the engines early life, the RUL decreased linearly but the sensor values remained constant up until the point that the error condition occurred at which point the sensor values would deviate from normal. This meant that the models were largely being trained to predict a changing target value based on constant input data. To minimise this a maximum RUL value of 85 was imposed on the training data so that the RUL would be a constant value up until closer to the end of the engines lifetime when an error was likely to occur. This improved score on the RUL prediction models but not on the label1 and label2 classification models.

It was interesting to see the performance of the LSTM network using a sequence of raw sensor data inputs compared to the other models using a single time instances with the aggregated features added.

A challenging part was determining a cost metric that would reflect the costs associated with incorrect predictions.

Improvement

Ways in which the predictive maintenance solution could be improved are:

- A neural network with more layers or more units per layer along with experimenting more with different sequence lengths could improve the score
- Using more engineered features such as different window sizes, differences between sensor values, variation from an initial or mean value, velocity of change could improve the score
- Trying more models such as SVM's could provide better results
- Using more techniques which are described in research papers around similar problems such as ways of splitting the dataset for training and validation, various cost functions, ways of removing noise in the sensor values, ways of detecting variations in signals, etc could be used.

- Further experimenting with setting the max RUL such as recognising the cycle at which the sensor values start to diverge from normal operating conditions could improve results