Machine Learning Engineer Nanodegree

Capstone Project

Predictive Maintenance

Ross Green
March 21st, 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.

This project explores a predictive maintenance solution on a simulated engine dataset. 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.

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 much longer an in-service asset has 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.

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:

Mean of:

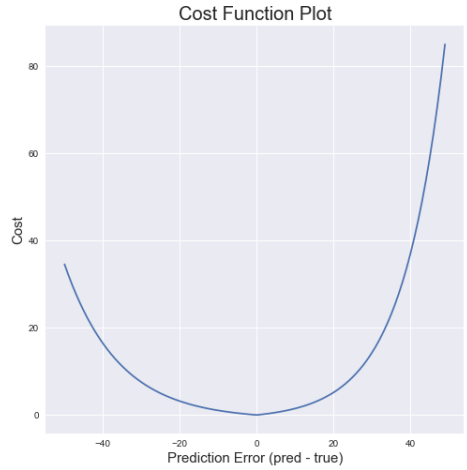
If pred – truth < 0:

$$e^{-\frac{pred-truth}{14}} - 1$$

Else If pred – truth > 0:

$$e^{\frac{pred-truth}{11}} - 1$$

which gives the plot below



To capture the cost for label 1 I made the cost of predicting 0 (wont fail within w1 cycles) when the actual label was 1 (will fail within w1 cycles) equal to 2, while predicting 1 when the actual label was 0 equal to 1.

To capture the cost for label 2 (2: fail within w0 cycles, 1: fail within [w0, w1] cycles, 0: wont fail within w1 cycles), I made the cost equal to 3 times the absolute difference if the prediction minus the actual was negative e.g. if prediction was 0 but the actual was 2, then the cost would be 3 * 2 = 6. If the prediction minus the actual was positive then the cost is the difference, e.g. if the prediction was 2 but the actual was 0 then the cost is 2.

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

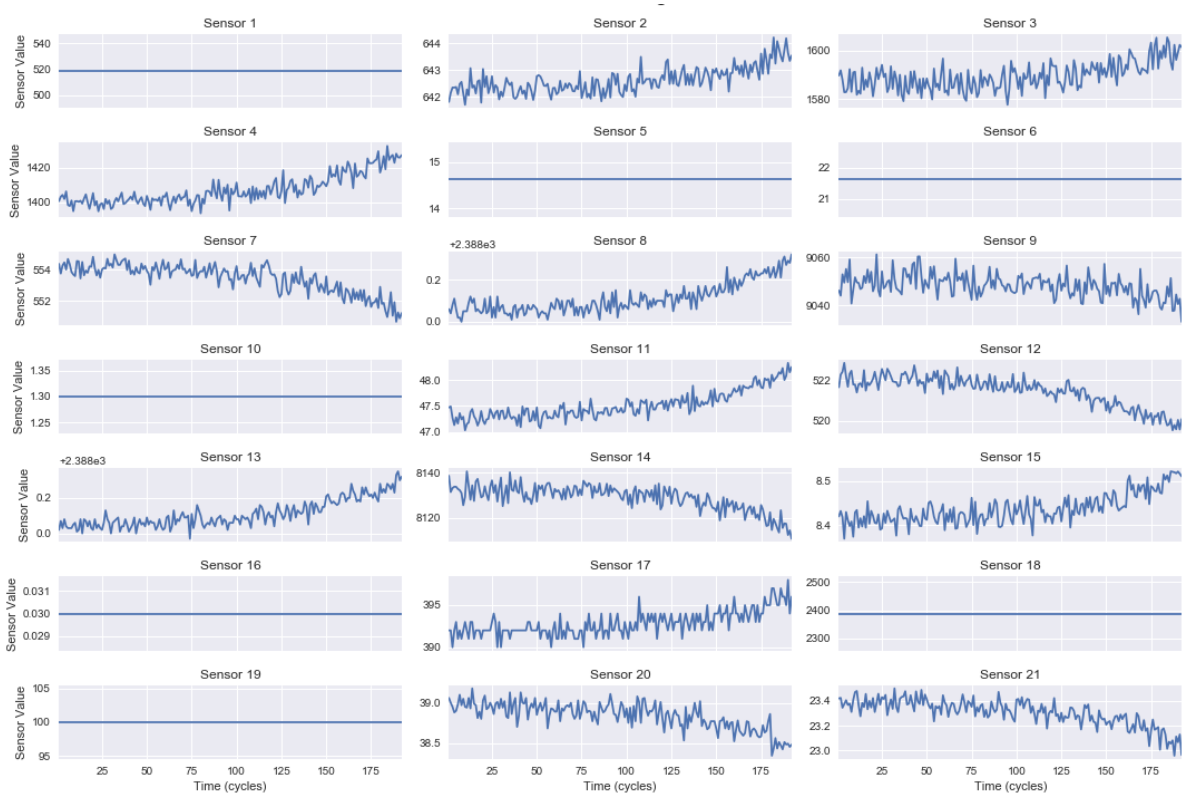
100 rows

RUL
112
98
69
82
91

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

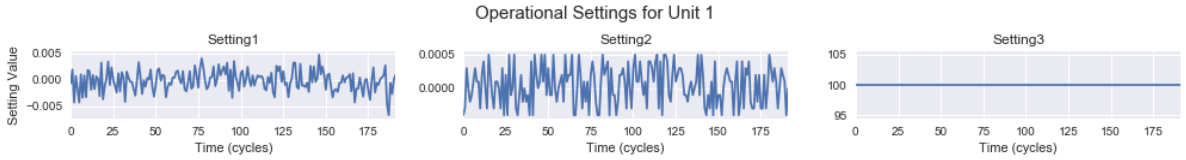
The training dataset provides data up until the the engine fails, so the last cycle is when the failure event occurs. The test data only provides a window of the assets operational lifetime and gives no indication of when the failure event occurs. The ground truth dataset provides the number of cycles 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 Visualization

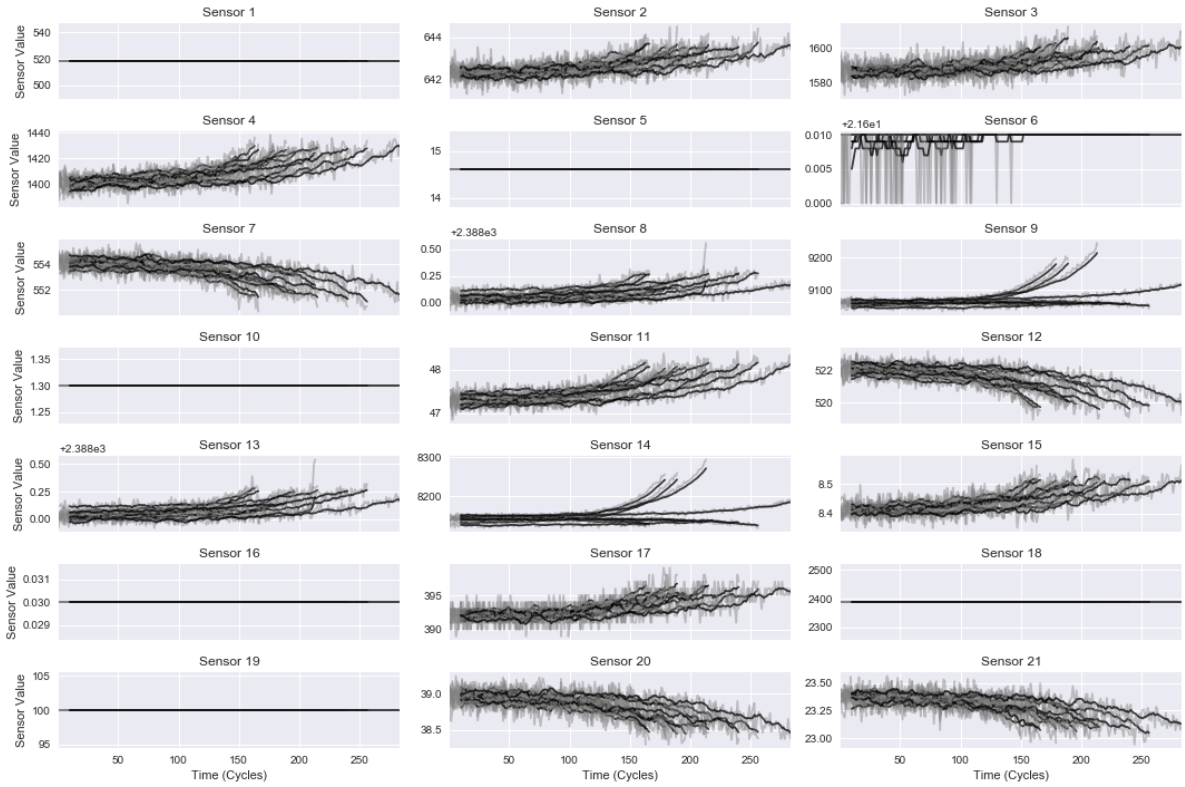


The figure above shows the sensor readings for engine unit 1 over its lifetime (192 cycles). We can see that some sensors increase over time, some decrease and some don't change at all. We can also see that each sensor value contains some amount of noise.

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

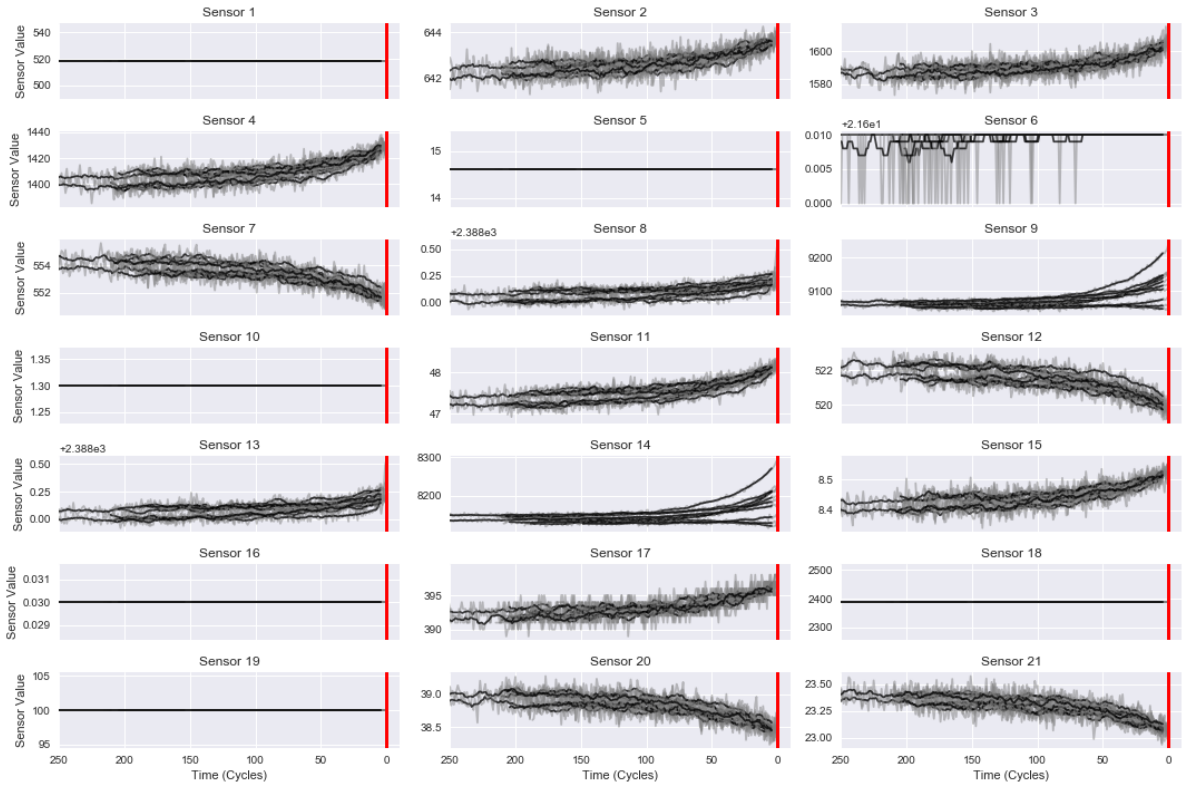


Each engine starts with a different level of wear and is run until failure. The training example contains the sensor values at each cycle up until failure. The figure below shows the sensor values from a random selection of 10 different engine units against time.



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.

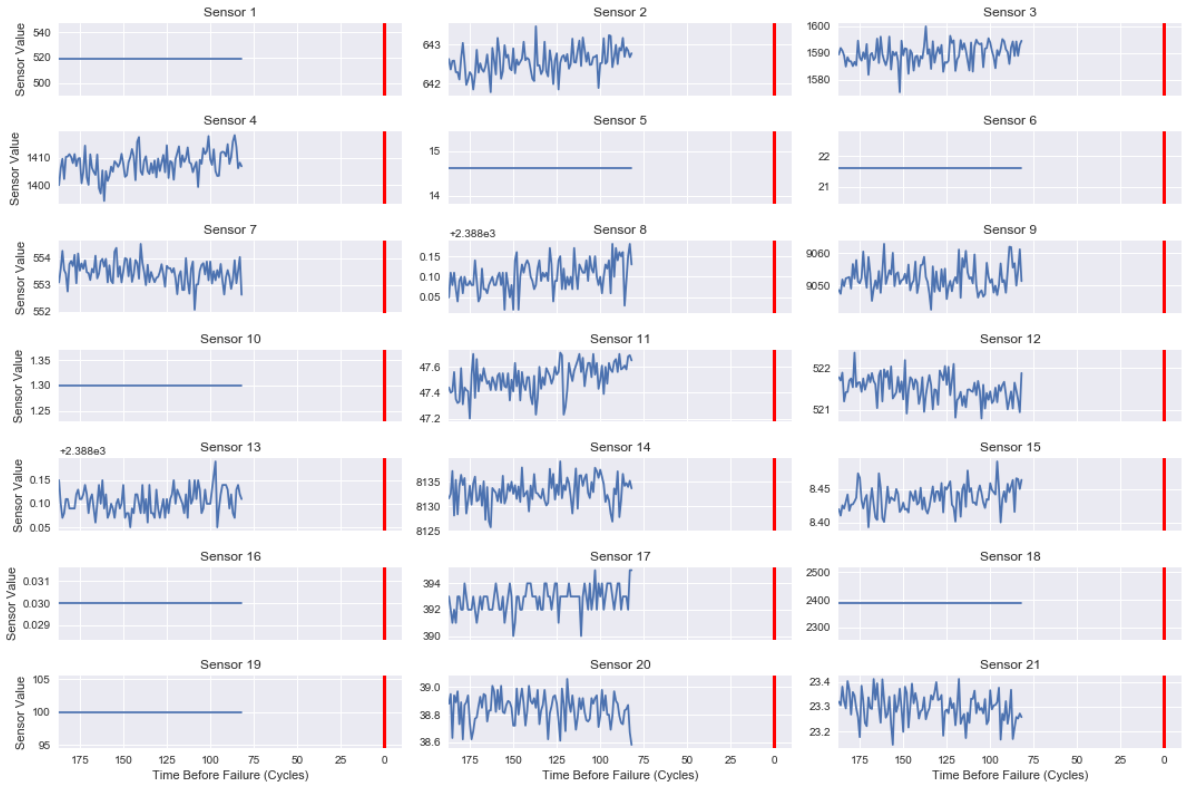
Since we know the cycle that each engine unit fails on, we can compute the number of cycles until failure at each time step, which is the engines lifetime minus the the current cycle. The figure below shows the engines sensor readings plotted against its time before failure. Each engine stops at cycle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

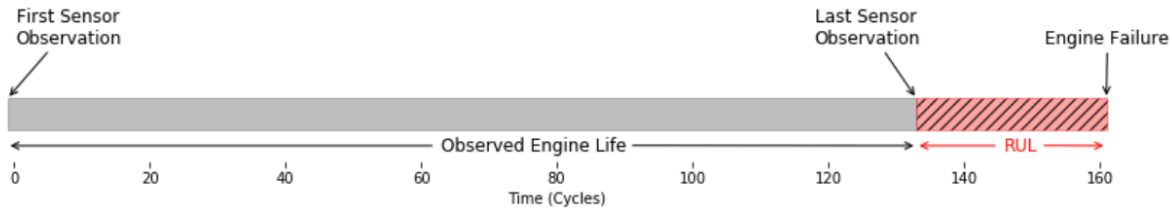


In this figure we can observe some patterns, such as some sensors (Sensor 4) rising before failure, some sensors (Sensor 7) falling before failure, and some (Sensor 9 and 14) having different behaviour for different engines.

The Prediction Challenge

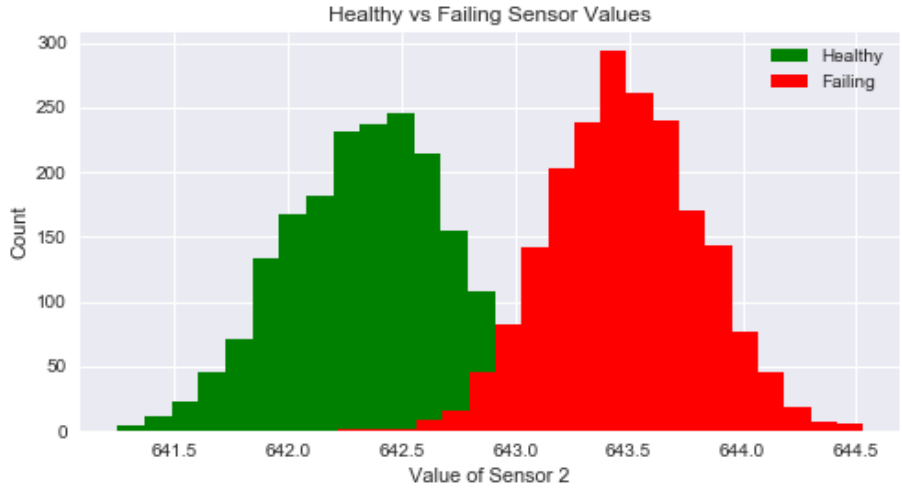
In the test set each engines measurements are truncated some unknown amount of time before it fails. The figure below shows the sensor readings for 10 randomly selected engines plotted against their time to failure. After observing the engines sensor measurements for a certain amount of time, the challenge is to predict the amount of time (number of cycles) the engine will continue running before it fails, which is the number of cycles until the failure point indicated by the red line below. This time is the Remaining Useful Life (RUL) of the engine.



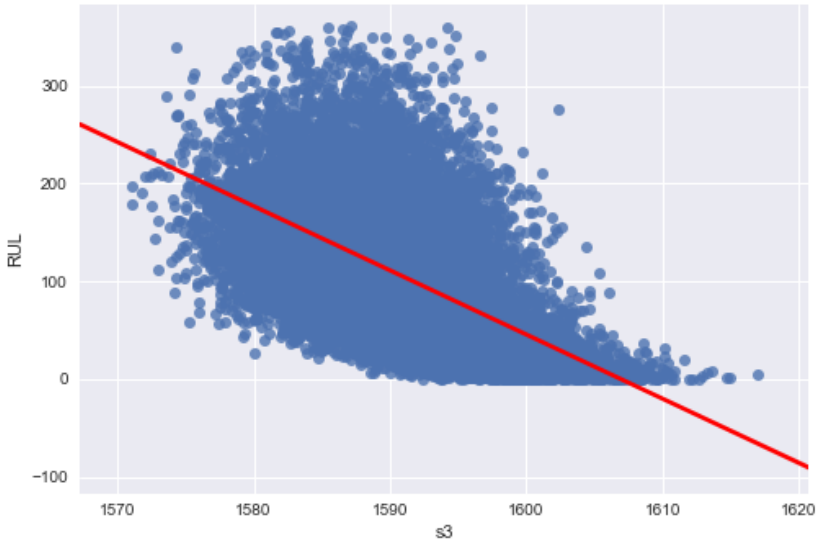


Most of the sensor readings change over the course of the engines lifetime and appear to exhibit consistent patterns as the engine approaches failure. The challenge of predicting the remaining useful life of an engine relies upon the engine readings having enough information to distinguish the state of the engine, for example whether it is healthy or approaching failure.

The figure below shows the distribution of sensor values of engine unit 3 and engine unit 7 in their healthy (first 20 cycles) and failure (last 20 cycles) states.



It is apparent that the distribution shifts from left to right for sensor 2 and right to left for sensor 7. The difference in distributions indicates that a model trained on the sensor data should be able to distinguish the state of the engine between the beginning and end of its lifetime.



A simple approach to predicting the RUL is by using a linear combination of the sensor values. The figure above shows the values of Sensor 3 plotted against the RUL value at each time cycle for every engine in the training set, with the line of best fit in red.

Algorithms and Techniques

The RUL is a continuous numerical variable which represents the number of operating cycles the engine has before it fails and is therefor a regression problem.

Label 1 is predicting one of two classes, namely 0 if the engine wont fail within a designated time window and 1 if it will. This is a binary classification problem.

Label 2 is predicting if the engine will fail in one of multiple time windows and is therefor a multiclass classification problem.

For each category, I use a linear model: Linear Regression to predict RUL and Logistic Regression models for label 1 and label 2, a Random Forrest Regressor / Random Forrest Classifier and a LSTM Deep Neural Network.

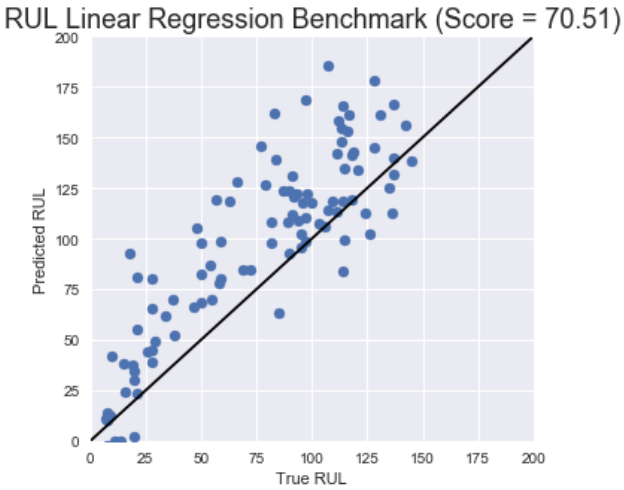
Random Forrest was chosen due to its flexibility in learning non linear patterns and its ability to reduce overfitting.

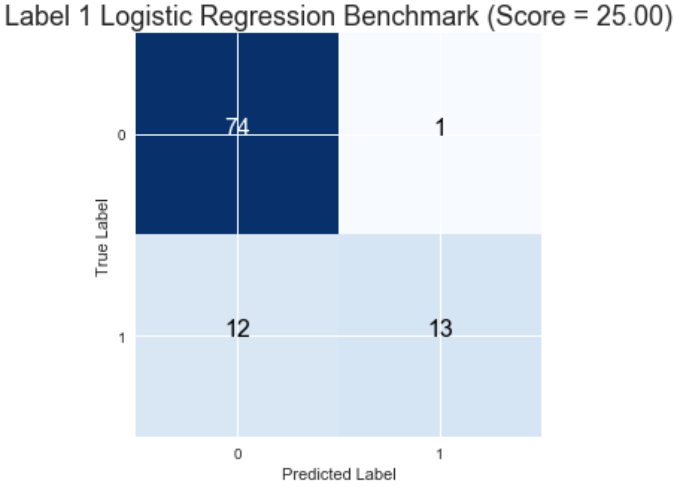
The hyper-parameters are optimized using cross validation with the training data split into 5 groups made up of 20 engines each. Then the model is trained on 4 of the groups or 80 engines and validated on the remaining group, and the process is repeated 5 times so each group is used as in the training set and as a validation set. The parameters are tuned using RandomizedSearchCV to try a number of different parameter combinations and choose the best one based on the average performance on the validation sets.

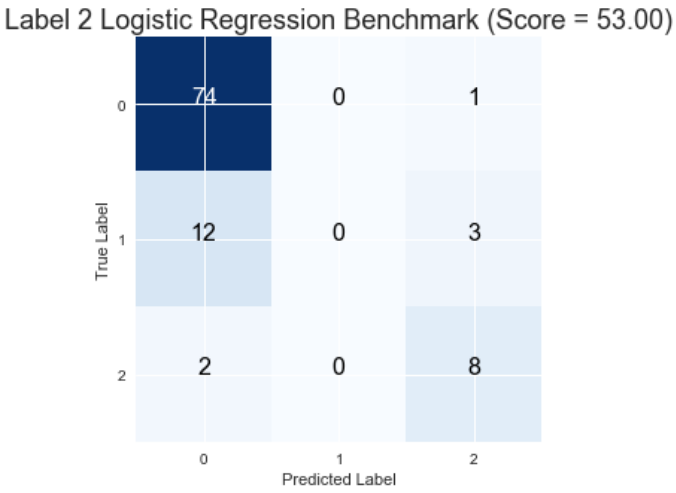
LSTM's are particularly attractive for sequence based (time series) data since they can automatically extract the right features from the given sequence of data, removing the need for manual feature engineering. Patterns that exist in the data sequence should be encoded in the LSTM network.

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.







III. Methodology

Data Preprocessing

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.

I then filter the sensor values to use by choosing only the sensors that have any variance over the engines lifetime, in other words, the sensors that remain constant are discarded since they don't provide any information about the engines health.

From the chosen sensor values I then create some aggregated features which summarise the historical activity of each asset. These aggregated features include the rolling mean, and rolling standard deviation of the *w* most recent values at each cycle.

When using the LSTM model, the data is first normalised using a MinMaxScaler to transform every value to a range between 0 and 1.

Implementation

The training data is made from the raw sensor values and the aggregated features along with the target value for each cycle. The test data is made from the last cycle for each engine totalling 100 rows.

The LSTM dataset is made using just the raw sensor values and target value which is turned into sequences of length 50, resulting in a 3D dataset with dimesnsions [number_sequences, sequence_length, number_features].

Each model is trained and its performance is measured with the respective cost function as defined in Metrics.

Each LSTM network contains 100 an LSTM layer made up of 100 units, followed by a dropout layer set at 0.2, followed by another LSTM layer made of 50 units and another dropout layer set at 0.2.

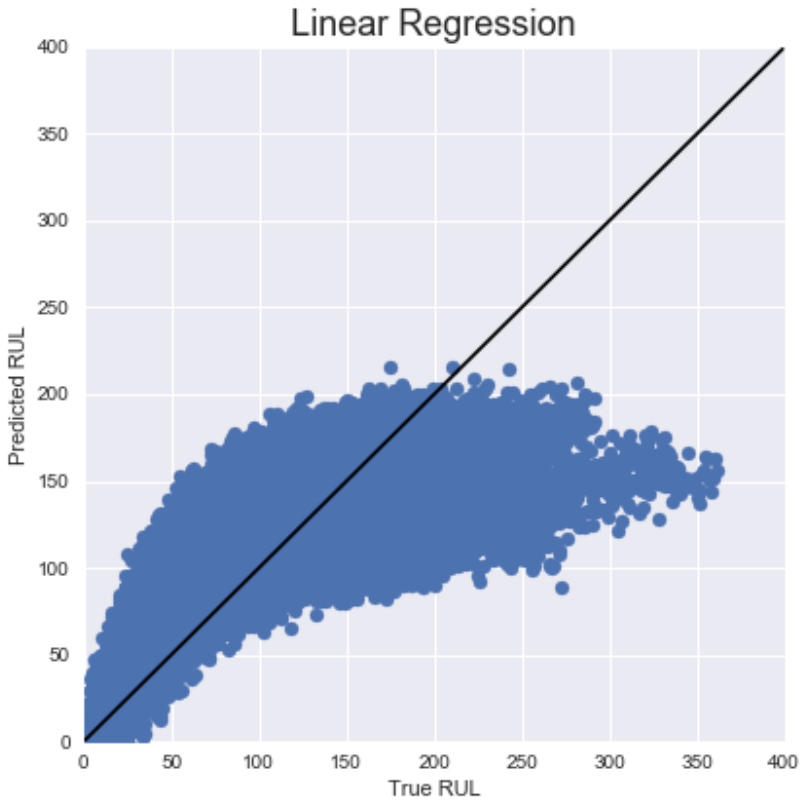
The RUL prediction network uses a fully connected Dense layer made of 1 unit with a linear activation. Since the RUL target value was normalised, the predicted value will be a value between 0 and 1. The predicted value in then inverse transformed back to the original range.

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.

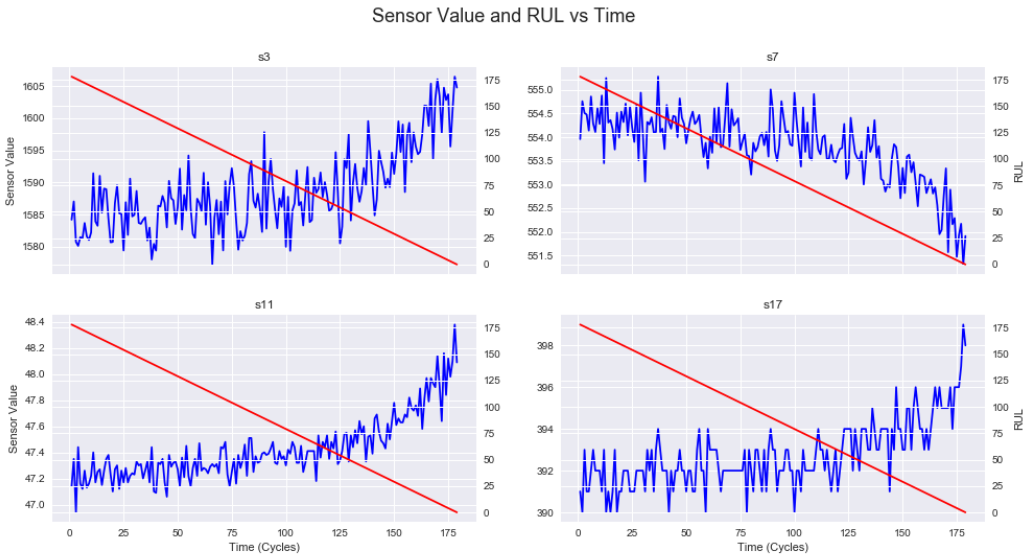
The label 2 prediction network has a Dense layer made up of 3 units with softmax activation. This gives the probability distribution of belonging to each of the 3 classes. The output with the highest probability is chosen as the predicted class.

Refinement

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).



Here we can see that the model is able to predict the RUL reasonably well when the actual RUL is below a certain threshold (around 100). Beyond this the predicted RUL remains fairly consistent and mostly below 200, while the actual RUL increases beyond 200.

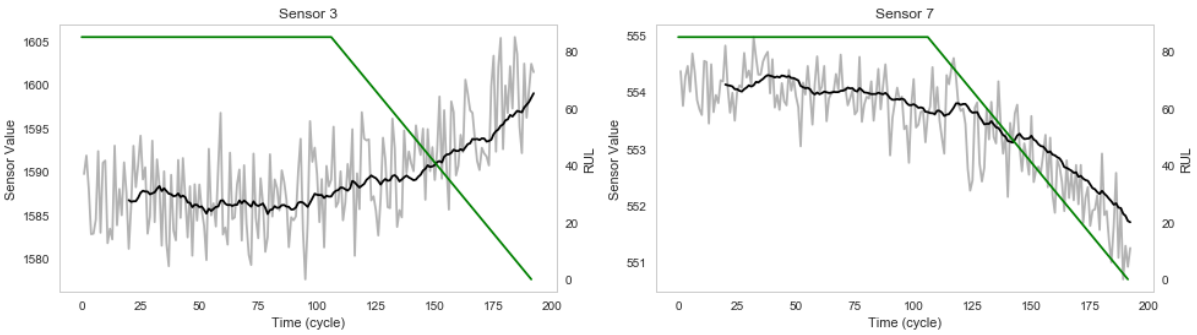


Looking at the above figure, we can see this makes sense, as the sensor values remain fairly consistent apart from noise up until an error condition occurs which causes the sensor values to deviate from normal, while the RUL value is linearly decreasing at each cycle. This means that the model is trying to predict a changing RUL value based on roughly consistent input sensor values.

If we think of RUL as a predictor 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 and then its RUL decreases, which is reflected in the sensor values.

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.

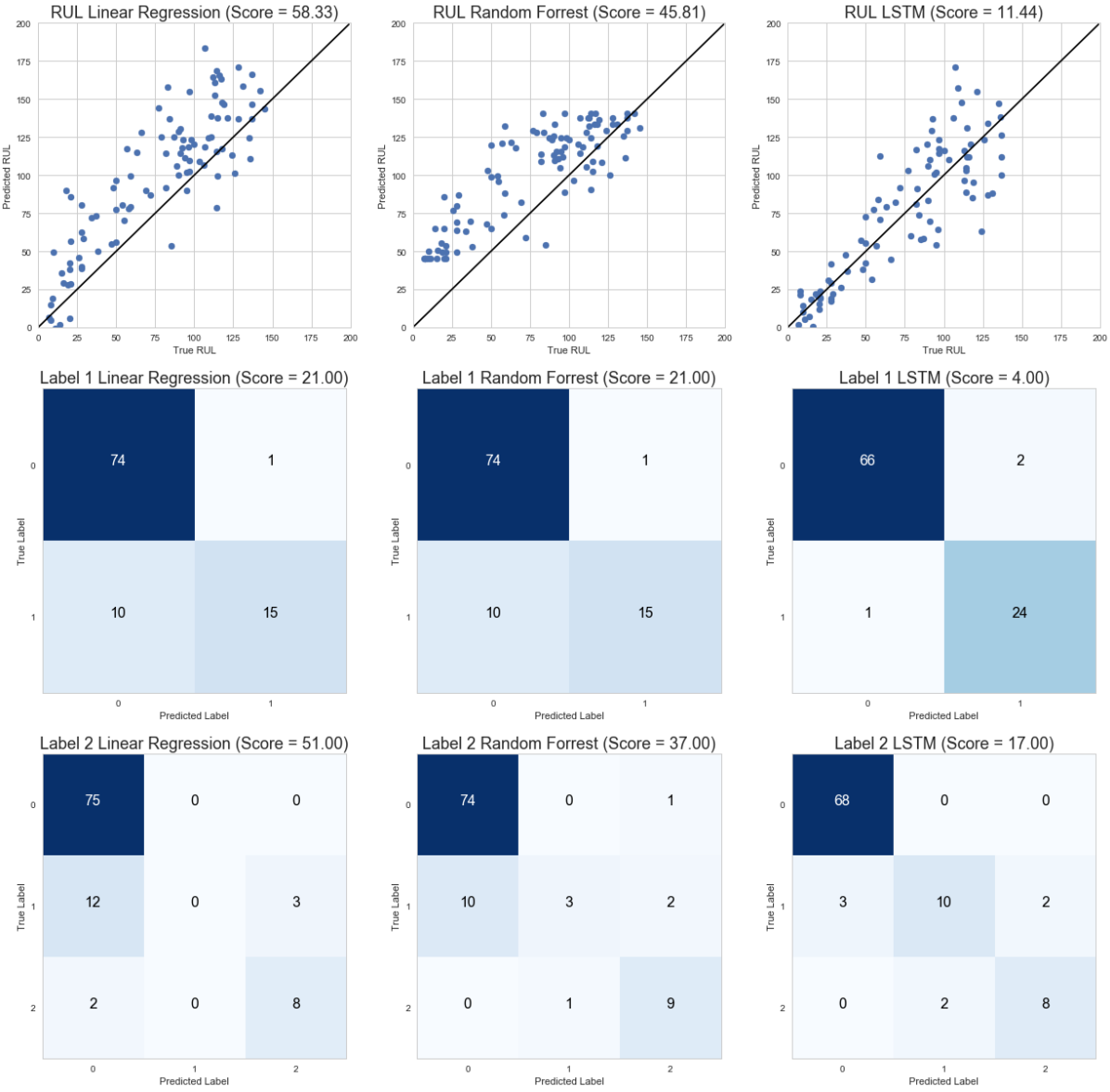


IV. Results

Model Evaluation and Validation

The results of the original models are summarised below.

	RUL	Label1	Label2
Benchmark	70.507539	25.0	53.0
Linear	58.326902	21.0	51.0
RandomForrest	45.809363	21.0	37.0
LSTM	11.443426	4.0	17.0



The results show that all models perform better than the benchmark model, with the LSTM network performing particularly well.

RUL:
Linear Regression and Random Forrest have a tendency to overestimate the RUL value, with most of the points falling above the line. The LSTM predictions follow the line much more closely, especially for the lower actual RUL values.

Label 1:

All models are good at predicting when a failure wont occur within 30 cycles. Only the LSTM model had good results when predicting that a failure would occur within 30 cycles.

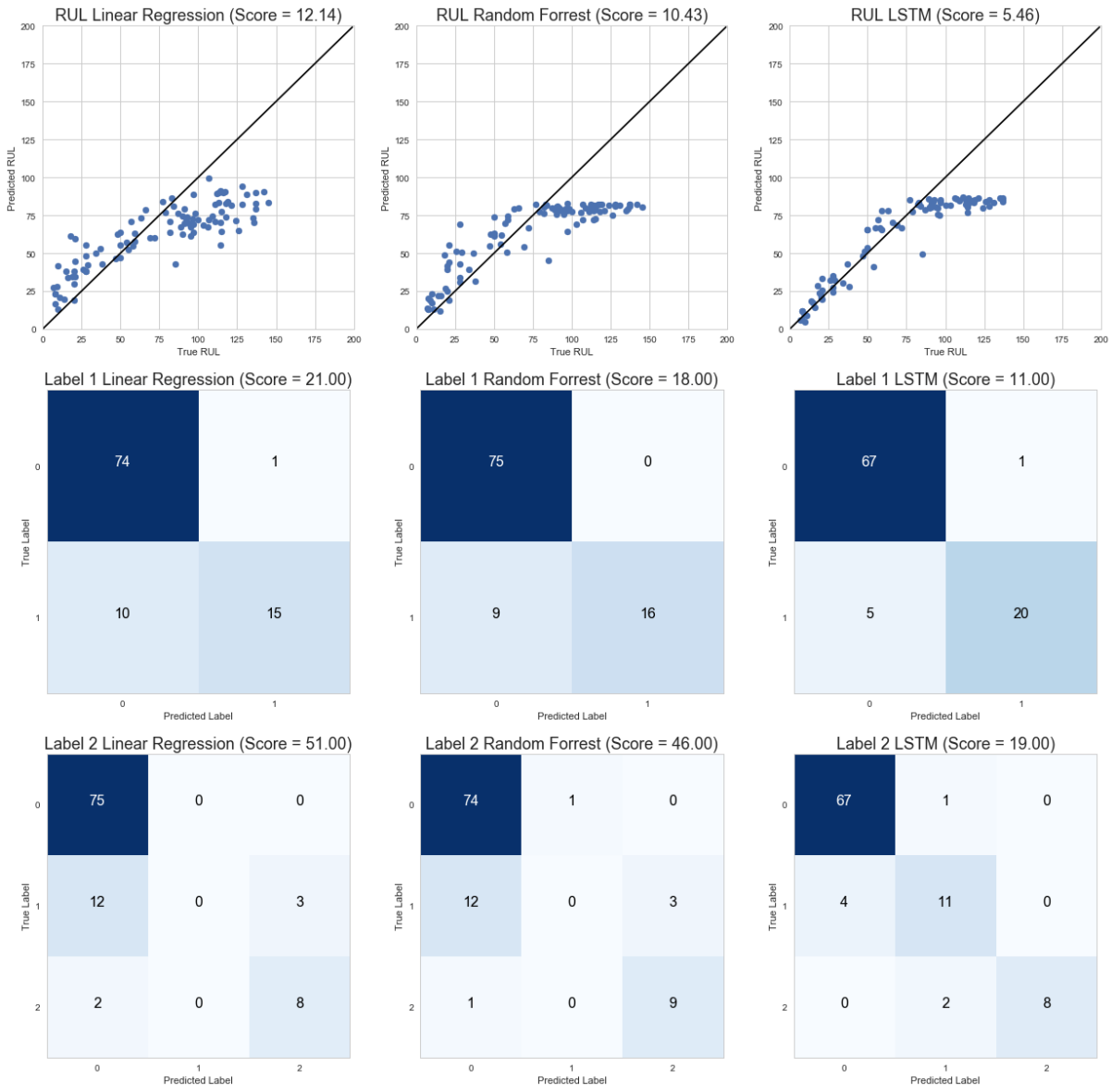
Label 2:

All models gave good results for predicting if an failure wont occur within 30 cycles (predict 0), or if a failure will occur within 15 cycles (predict 2). The models seemed to struggle when predicting if a failure would occur in the 15 to 30 cycle window. Again the LSTM network did a pretty good job, while the other 2 were very inaccurate.

After Refinement:

The results shown below are from the adjusted dataset where the RUL is set to a maximum value of 85.

	RUL	Label1	Label2
Benchmark	70.507539	25.0	53.0
Linear	12.138950	21.0	51.0
RandomForrest	10.430531	18.0	46.0
LSTM	5.457015	11.0	19.0



These results show a decent improvement in RUL predictions score for all models, 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. The classification results on the other hand did not show improvement except for a slight improvement in the Label 1 Random Forrest Score and the accuracy of predicting Label 2 ‘1’ class actually got worse for all models with more

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 prioritizing 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.

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

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.

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