PREDICTIVE MAINTENANCE WITH MACHINE LEARNING

Project Final Report

1. Domain Background

In any asset heavy industry like manufacturing, utilities, energy generation and distribution, maintaining the optimal health of the assets and machines is a critical aspect of the operations. If the assets are not in healthy state it not only impacts the performance and productivity, but may also lead to serious HSEC issues (health, safety, environment and compliance)

Maintenance professionals have always faced this conundrum of creating effective maintenance strategy and planning, with a target to maximize the uptime and minimize the downtime whilst maintaining the healthy state of machines. It can be quite difficult to determine how often and when a machine should be taken offline to be serviced as well as weight the risk of machine downtime while attempting to maximize the uptime through early maintenance and replacements

Traditionally various techniques have been employed – quantitative and qualitative in order to identify the fault modes and plan the maintenance of the various machines in the manufacturing facilities.

Components of maintenance program often fall in one of the four categories below:

- Reactive Maintenance
- Planned Maintenance
- Proactive Maintenance
- Predictive Maintenance

In this capstone project we have explored how to effectively undertake the journey from the paradigm of **Preventive Maintenance** (scheduled, time-based maintenance) to the paradigm of **Predictive Maintenance**, where we are able to predict when a failure may occur based on the telemetry data coming from variety of the sensors on the equipment and machinery.

2. Project Introduction

The sensor telemetry data from the equipment and production assets is sensitive and is not freely available, for this reason in this project I have relied on the **Turbofan Engine Degradation Simulation** dataset provided by the NASA's prognostic centre. The dataset and the details of the simulation setup can be accessed from the following link:

http://ti.arc.nasa.gov/project/prognostic-data-repository

It is assumed that the sensor and telemetry data from the air-turbine have hidden patterns related to engine's health and condition, and thus can be used the predict the **TTF (time-to-failure)** or **RUL (remaining-useful-life)** for an engine. This way the maintenance activities can be planned based on the TTF predictions complementing the time-based maintenance activities (and eventually replacing this approach for full-fledged predictive maintenance)

In this Project Report I have summarized the work done and the results of using Exploratory Data Analysis and machine learning techniques to acquire, process and analyse the sensor measurements and the prediction results generated by different machine learning algorithms.

I have also briefed how we can use machine learning predictions to come up with the Cost Benefit analysis and facilitate the informed (data-backed) business decisions.

3. Approach

Whilst the simulation dataset has the telemetry data leading to run-to-failure events, but generally data driven prognostic scenario faces the challenge of the lack of data for run-to-failure events. Most of the time the data contains the fault signature for growing fault pattern, but no or very little data capture of fault evolution until failure.

Hence as a first step I have ignored the run-to-failure event data and performed following activities:

> - Identify clusters in the training dataset,
> - perform Principal Component Analysis to visualize the clusters
>
> Additionally, I have discussed an approach of how we can gradually move from preventive maintenance to predictive maintenance, by overlaying the scheduled maintenance timelines on the clusters (normal, warning, critical) of engines in the training data

I have then utilized the run-to-failure event data and then utilized variety of supervised learning algorithms to make the following predictions:

> - Use of regression algorithms to predict TTF (time-to-failure)
> - Use of binary classification algorithms to predict if engine will fail in the current period
> - Use of multi class classification algorithms to predict in which period an engine will fail.

We have assumed the period to be 30 cycles for the experiment purposes, but in actual this will dependent on the scenario and context of the operations

The project source code is available in the following GitHub repo:

https://github.com/sanjaysoni2k1/Predictive-Maintenance

In the next few sections I have summarized the activities performed and the results of exploratory data analysis and machine learning activities.

4. Dataset: Data Acquisition

The simulation dataset contains training, test and the ground truth data for 4 different groups – FD001, FD002, FD003 and FD004. For our project we have used the training, test and ground truth data for FD001 group.

In FD001 data series

- engines are operating normally at the start of each time series.
- engines have started operating at different point in time before the data capture activities for FD001 started.
- engines develop a fault at some point during the series and is the last cycle in the series.

The summary of the individual data file is as follows:

- Training Data: contains the run-to-failure data for 100 engines, and 20,000+ cycles (across the 100 engines)
- Test Data: contains the operating data without the failure events
- Ground Truth Data: contains the true remaining cycles for each engine in the test data

The data in the training and test data series contain following information:

- **id:** the engine id ranging from 1 to 100
- **cycle:** per engine.
    - starts from 1 and to cycle number where failure occurred (in training data)
    - starts from 1 to some random cycle number before failure occurred
- **settings 1 to settings 3**: engine's operational settings
- **s1 to s21**: sensor measurements

The ground truth data contains the RUL / TTF data for all the 100 engines in the dataset.

Data snapshots is provided in the screenshots below:

Ground Truth Data Snapshot:

	rul
0	112
1	98
2	69

Training Data Snapshot:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3	47.49
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3	47.27

Test Data Snapshot:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
0	1	1	0.0023	0.0003	100.0	518.67	643.02	1585.29	1398.21	14.62	21.61	553.90	2388.04	9050.17	1.3	47.2
1	1	2	-0.0027	-0.0003	100.0	518.67	641.71	1588.45	1395.42	14.62	21.61	554.85	2388.01	9054.42	1.3	47.5
2	1	3	0.0003	0.0001	100.0	518.67	642.46	1586.94	1401.34	14.62	21.61	554.11	2388.05	9056.96	1.3	47.5

5. Data Wrangling

On performing initial checks on the dataset following observations were made:

- Data type for all the feature columns are numeric
- There are no null values, or missing value in the feature columns.

Training Dataset

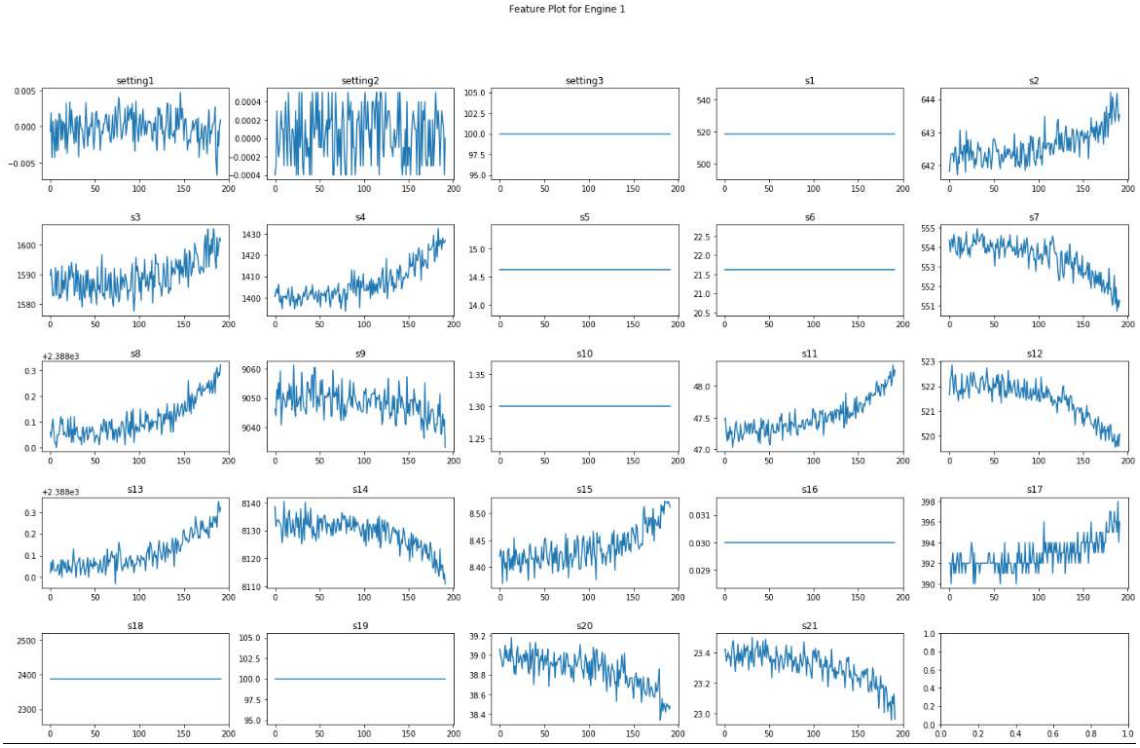
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20631 entries, 0 to 20630
Data columns (total 26 columns):
id          20631 non-null int64
cycle       20631 non-null int64
setting1    20631 non-null float64
setting2    20631 non-null float64
setting3    20631 non-null float64
s1          20631 non-null float64
s2          20631 non-null float64
s3          20631 non-null float64
s4          20631 non-null float64
s5          20631 non-null float64
s6          20631 non-null float64
```

Test Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13096 entries, 0 to 13095
Data columns (total 26 columns):
id          13096 non-null int64
cycle       13096 non-null int64
setting1    13096 non-null float64
setting2    13096 non-null float64
setting3    13096 non-null float64
s1          13096 non-null float64
s2          13096 non-null float64
s3          13096 non-null float64
s4          13096 non-null float64
s5          13096 non-null float64
```

We also analysed all the sensor data and operational settings telemetry for engine 1 to get an intuition of the data and trends and figured out that sensors: **s1, s5, s6, s10, s16, s18, s19** and operational setting **'setting3'** are constant features.

On further analysis we found out that these features are almost constant with very little variations for all the engines and thus concluded that including these features as input to machine learning algorithms will have no benefits and may infact impact performance of the models.



	count	mean	std	min	25%	50%	75%	max
setting3	20631.0	100.000000	0.000000e+00	100.00	100.00	100.00	100.00	100.00
s1	20631.0	518.670000	6.537152e-11	518.67	518.67	518.67	518.67	518.67
s5	20631.0	14.620000	3.394700e-12	14.62	14.62	14.62	14.62	14.62
s6	20631.0	21.609803	1.388985e-03	21.60	21.61	21.61	21.61	21.61
s10	20631.0	1.300000	4.660829e-13	1.30	1.30	1.30	1.30	1.30
s16	20631.0	0.030000	1.556432e-14	0.03	0.03	0.03	0.03	0.03
s18	20631.0	2388.000000	0.000000e+00	2388.00	2388.00	2388.00	2388.00	2388.00
s19	20631.0	100.000000	0.000000e+00	100.00	100.00	100.00	100.00	100.00

As we can see the minimum and maximum value for these sensors is constant. We will thus ignore these sensor readings from our analysis

Feature engineering (feature extraction) was also applied to the training and test data, time domain features – rolling standard deviation and rolling mean was added for each of the 21 sensor columns. The rolling average features also have the smoothening effect on the original feature data as a result of which any noise in the data is reduced. This was apparent when the mean feature data is plotted on the original (noisy feature data).

Then the regression and classification labels for training data were created as follows:

Regression Label: Time-to-Failure (number of remaining cycles before failure) – the last cycle for each engine in the dataset was calculated, and for each record per engine difference between current cycle and last cycle was captured as TTF

Binary Classification: if the remaining cycles is less than the specific period i.e., number of cycles (period = 30 was selected for experiment) then engine will fail in this period (label: 1) otherwise engine will not fail (label: 0)

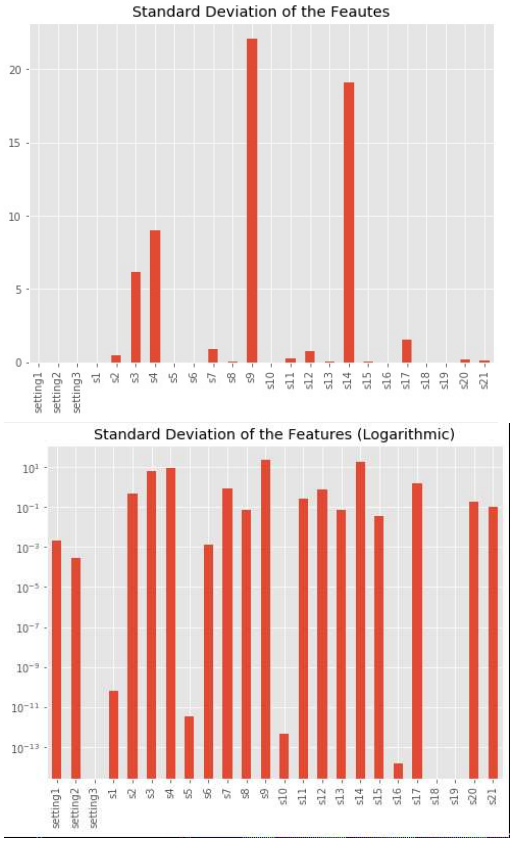
Multi-class Classification: the period of 30 cycles was divided into two groups thus leading to three different bands

- **0~15 cycles:** if remaining cycle is in this band – label 2 was assigned (critical condition i.e., maintenance activities should be carried over on the turbine)
- **15 ~ 30 cycles:** if remaining cycles is in this band – label 1 was assigned (warning condition i.e., turbine will need maintenance after next few cycles
- **30 + cycles:** if remaining cycles is in this band – label 0 was assigned which means that air turbine is in normal operation mode.

For test data, TTF is provided in a separate truth data file. The two files were merged then classification labels for test data were created the in the same way described above.

6. Features: Exploratory Data Analysis

Exploratory data analysis activities for the features at the macro level i.e. feature variability, distribution of the features and feature correlations were examined to uncover underlying structure and extract important variables.

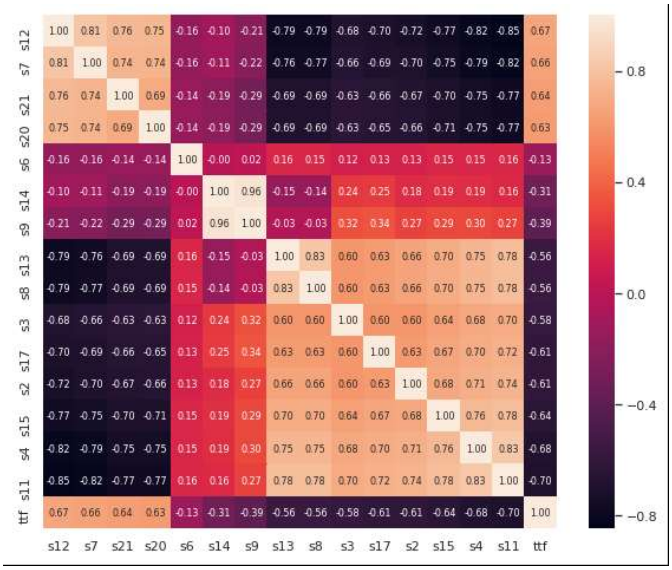


Feature	Standard Deviation
s9	22.0829
s14	19.0762
s4	9.0006
s3	6.1312
s17	1.5488
s7	0.8851
s12	0.7376
s2	0.5001
s11	0.2671
s10	0.1807
s21	0.1083
s13	0.0719
s8	0.071
s15	0.0375
setting1	0.0022
s6	0.0014
setting2	0.0003
s1	0
s5	0
s10	0
s16	0
s18	-
s19	-
setting3	-

As we can see the standard deviation of sensors – **s1, s5, s10, s16, s18, s19 and setting3** is 0 which is in sync with what we discovered during the initial data analysis in the previous sections.

In the first graph for standard deviation we can see that there is skewedness in the scale of standard deviations i.e. sum features show more variation whereas other show negligible variations. That's why we have plotted the standard deviations on the logarithmic scale in second plot.

As we can see from the table above some of the features have very low variation (std <0.2) – ***(s10, s21, s13, s8, s15, s6, setting1, setting2).*** These can be the contenders for reducing the feature space (in case there is large number of features).



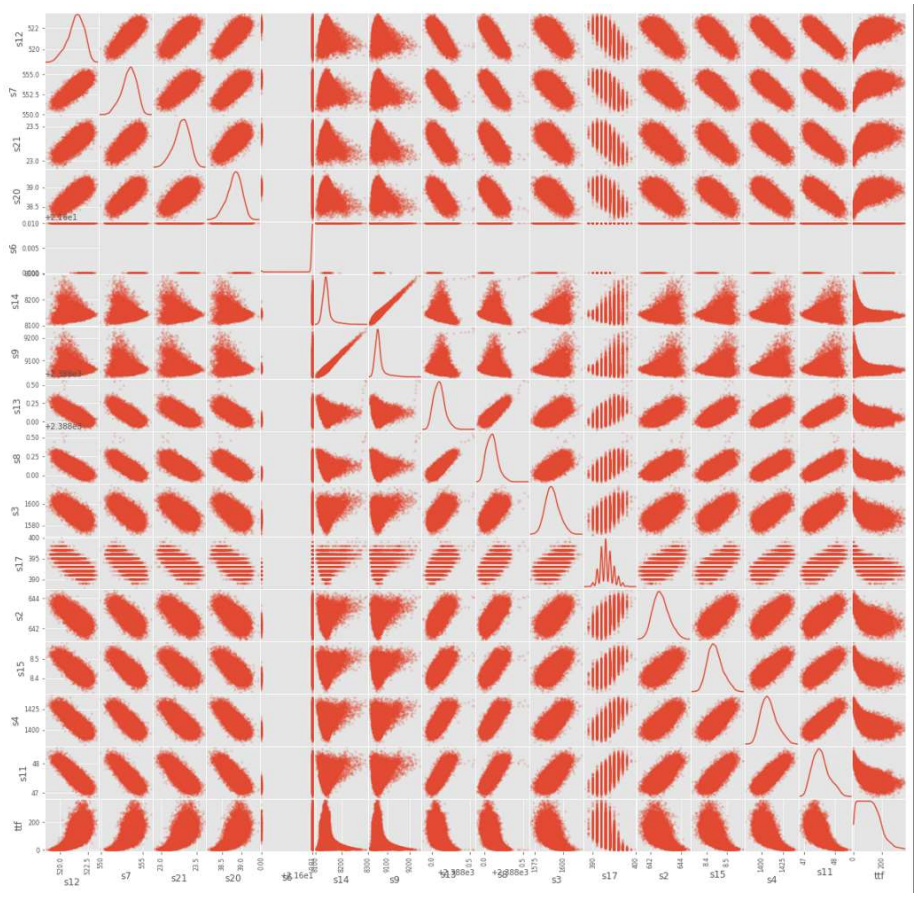
Features with high variability were checked for correlation with other features and regression label.

Heatmap analysis revealed that there are pair of features with high degree of correlation (>0.8). Following pairs fall in this category: **(s14 & s9), (s11 & s4), (s11 & s7), (s11 & s12), (s4 & s12), (s8 & s13), (s7 &s12)**

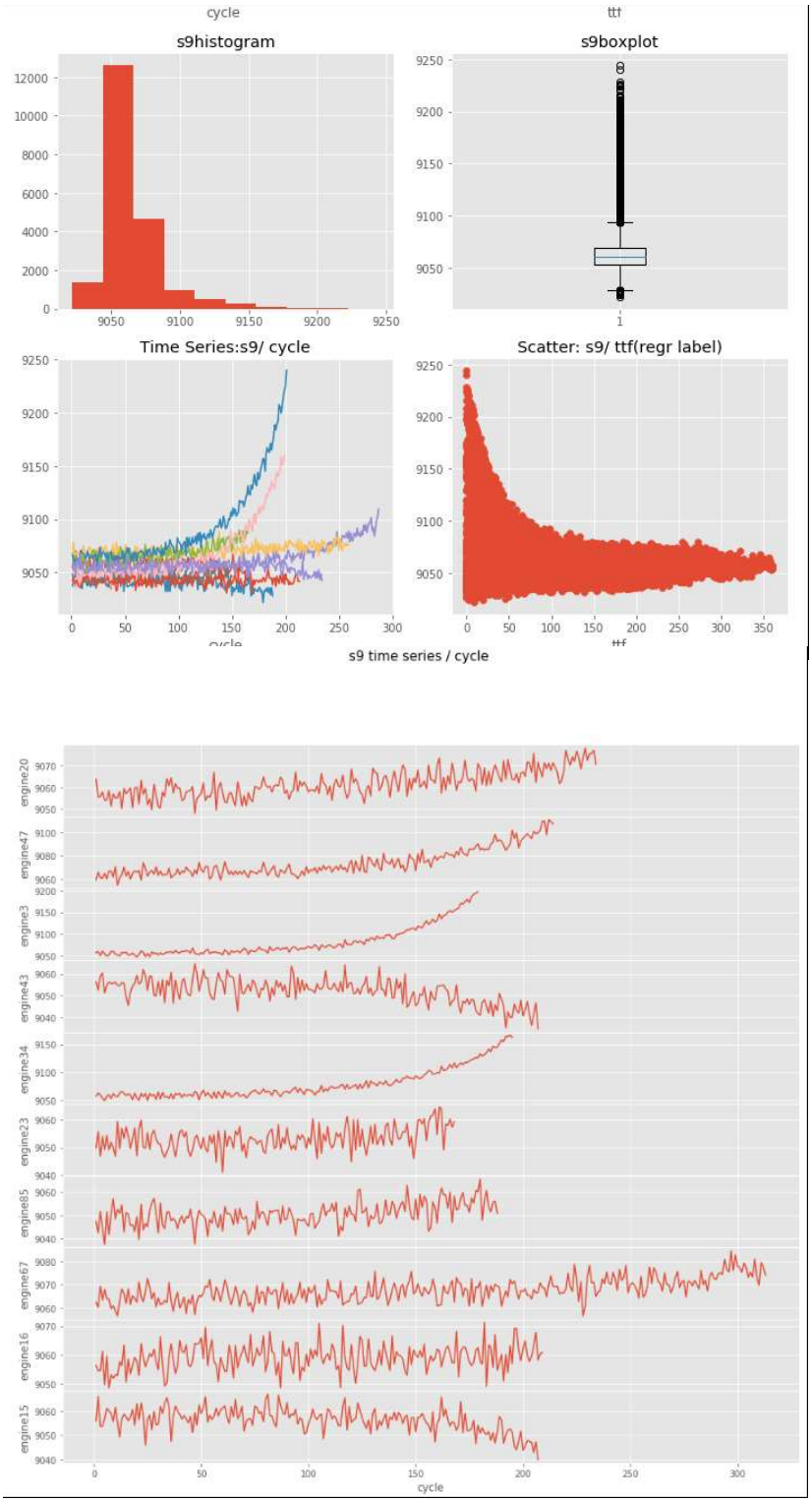
This multicollinearity may hurt the performance of some machine learning algorithms. So individual features from the above will be target for elimination during optimization

It's also observed that

- most features have non-linear relationship with the regression label RUL; hence the polynomial transformation of the features may enhance the model performance.
- most of the features have normal distribution, which is likely to improve the model performance.



A number of EDA charts were also used to generate more insight on feature individually, for e.g., s9



Amongst other things it tells us that there are outliers in almost all the feature data, data points which are outside the IQR range and may be considered for elimination to assist it data clustering and PCA

Plotting the sensor value with the cycles for every engine also graphically reveals how individual feature changes with the increase in cycle and that for different engines the cycle at which engine fails varies a lot.

For e.g. in the s9 time series / cycle plot below it is clearly visible that some engines have continued to operate for cycles more then 300, whereas some of the engines have failed as early as 150~165 cycles.

Thus we can expect to see some clustering in terms of when the engine is going to fail when data clustering is performed.

We can also observe that there is lot variation in the value range of the feature data so doing some scaling (Standard Scaler) will have positive impact on machine learning algorithms as well as for Principal Component and Cluster analysis.

Summary of Exploratory Data Analysis

- Following approaches may be used for feature reduction i.e. reduce number of features
    - Features with zero variation (i.e. zero standard deviation) can be removed - s**1, s5, s10, s16, s18, s19 and setting3**

    - Features with variation less than a threshold (say 0.2) can be removed if required - ***s10, s21, s13, s8, s15, s6, setting1, setting2***

    - There are feature pairs with high degree of correlation amongst themselves (multi collinearity) - **(s14 & s9), (s11 & s4), (s11 & s7), (s11 & s12), (s4 & s12), (s8 & s13), (s7 &s12)**

      One feature each from these pairs can be removed to reduce the number of features and also improve machine learning performance.

- There are outliers in almost all the feature data series. We can look to pre process the data to remove the outliers to improve the performance of the machine learning models if required.

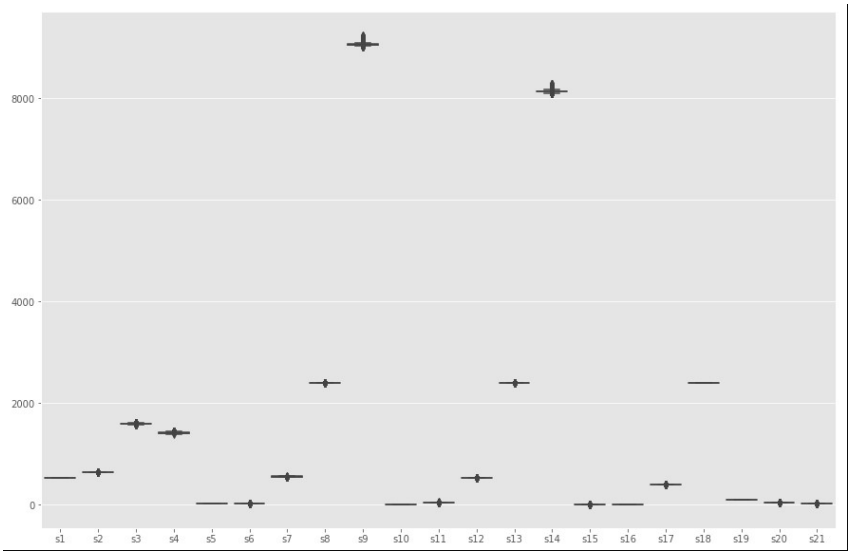
- We can also consider to perform the feature scaling as there is large variation in the value range of the individual features (this will assist the machine learning to minimize the cost function faster)

- Our dataset has a class imbalance problem for binary classification and and multiclass classification labels – we only have around 20~30 % of data with Positive Predictions (i.e. air turbine will fail) and 70~80% of the data with Negative Prediction.

  So while chosing machine learning evaluation metrics – Accuracy will not be an appropriate measure and we should consider other metrics which takes into account False Negative as well. Evaluation approaches like Precision-Recall curve, F1Score, AUC-ROC should be considered

7. Principal Component Analysis and Data Clustering

We also analysed the range of minimum and maximum values across the feature space.

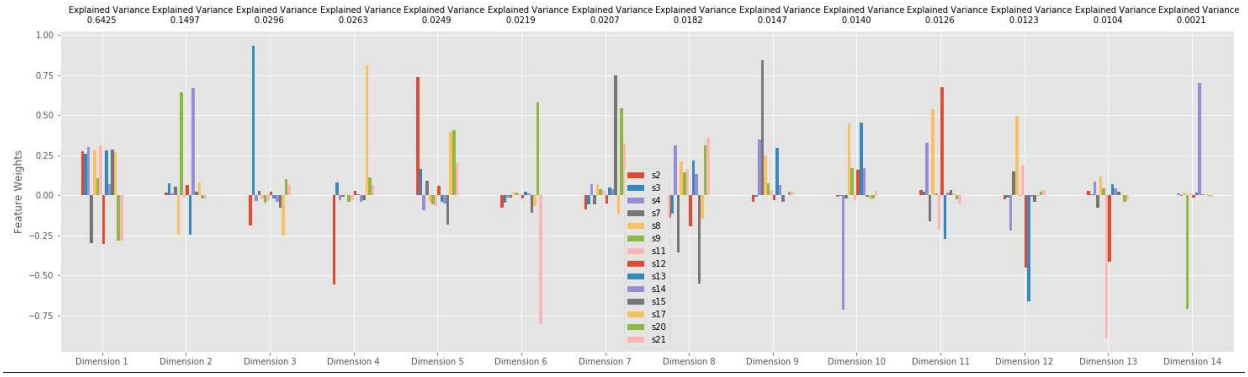


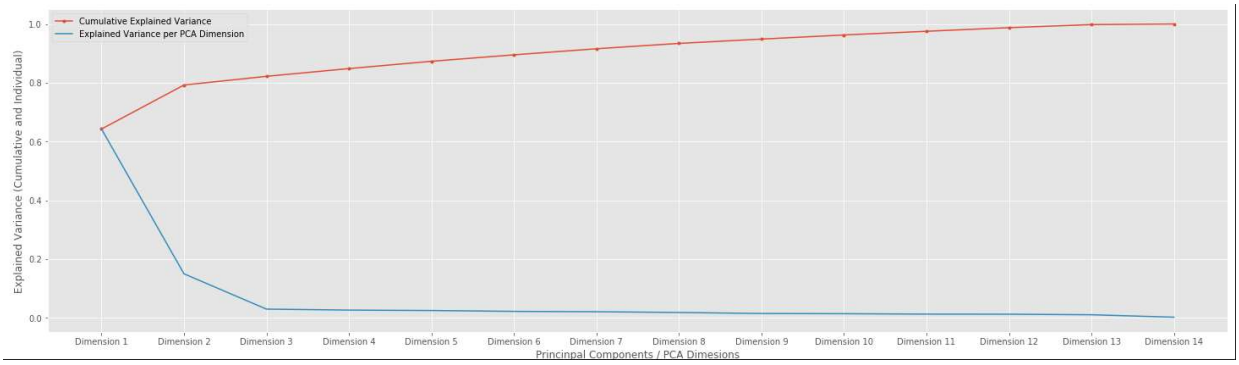
We can see that the range of the values (min and max) of the individual features and possible the variations that each feature exhibit is quite different.

Given these are sensor readings and based on the understanding of the data the sensors are measuring different attributes with different unit of measurements.

Feature scaling was performed for the purposes of Principal Component Analysis. PCA revealed that first Principal component accounts for 64.2% of the explained variance, and the second Principal Component accounts for another 15% of the explained variance.

Thus, the first two principal components itself account for 80% of the explained variance. Thus clustering using just two principal components should be able to identify the operational groupings (normal, warning, critical operational ranges) in the air turbine.

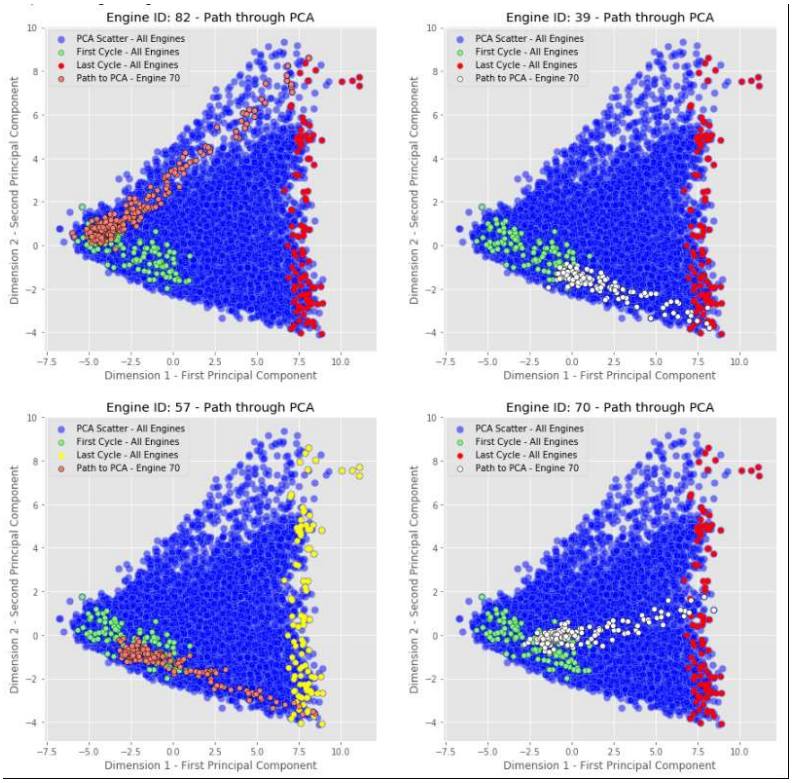




The scatter plot of the two principal components reveals that

- the last cycle of the engines varies a lot and is scattered across the edges of the plot (this the area of CRITICAL condition can potentially be isolated)
- the area of normal operations is clustered together, which is as per expected.

I also plotted that the path for some of the engines to see how they have traversed through the PCA feature space.



As we can see from the **path to PCA** plots:

- starting position of the cycles (i.e., the **normal operation** range) is clustered around the bottom left of the scatter plot.
- the end of the cycle (i.e. the **critical operation** range) is towards the edges of the scatter plot)
- we can assume a pseudo cluster in the center of the plot for our purposes, and classify that as **warning operation** range)

How to plan Predictive Maintenance using this information?

We can use this information to gradually (and conservatively delay the planned scheduled maintenance) and continuously observe the operational ranges of the engines.

For e.g., the original dataset also reveals that scheduled maintenance was performed after 150 cycles so if we overlay the above three clusters and start to gradually shift the planned maintenance say by +10 cycles and observing the individual engines as they shift from warning to critical state (threshold to be defined during operations) and update our clusters.

As we iterate through process, we will have

- sufficient fault patterns captured in the data
- revised and optimized operational ranges and threshold for normal, warning and critical operations
- as well as some run-to-failure events captured

With these data points captured we can move on the next stage of planning predictive maintenance where supervised machine learning approached can be applied to the telemetry data with run-to-failure events and optimized bands of operational ranges.

8. Regression Analysis – Predicting Engine's Time to Failure

To predict the engine's time to failure, is a regression machine learning problem. We tested various linear and non linear algorithm as part of the test harness, and performance metrics were calculated and evaluated for all the algorithm.

Following is the list of algorithms and its brief explanation:

Linear Regression: is the statistical method of analysis that estimates the relationship between one or more independent variables (features) and a dependent variable (label), by minimizing the cost function which by default is defined as the square root of the mean of the errors (difference between the predicted and observed value of the dependent variable) computed by a fitting a straight line or hyperplane.

Lasso Regression: is the variation of the linear regression in which the L1 regularization is performed i.e., by penalizing the L1 norms (sum of absolute value) of the coefficients

Ridge Regression: is a form of regression algorithm for analysing data with multiple collinearity. In this algorithm a degree of bias is added to the regression estimates thus reducing the standard errors

Polynomial Regression: is another variation of regression in which the relationship between the target variable 'y' and the features 'x' is modelled as N^{th} polynomial in x

Decision Tree: algorithm learn in a hierarchical fashion by repeatedly splitting the dataset into separate branches that maximize the information gain of each split.

Random Forest: is the ensemble technique in which the predictions from multiple decision trees, and the output is the mean of the prediction of the individual decision trees.

The following table show the results of the evaluation metrics for the algorithms.

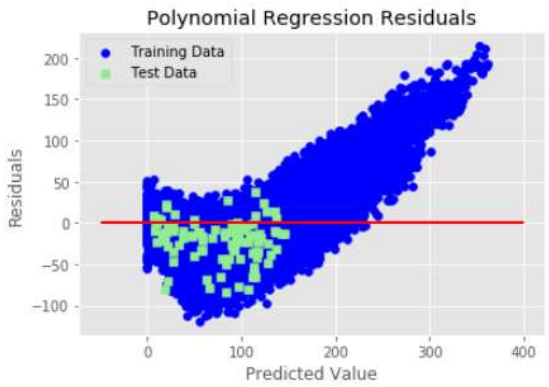
	Linear Reg	LASSO	Ridge Regression	Decision Tree Regression	Polynomial Regression	Random Forest Regression
Root Mean Squared Error	32.041095	31.966099	31.965740	32.095349	32.530030	28.634253
Mean Absolute Error	25.591780	25.551808	25.544620	24.319068	24.944453	23.167130
R^2	0.405495	0.408275	0.408289	0.403480	0.387213	0.525198
Explained Variance	0.665297	0.668206	0.667607	0.632767	0.633327	0.767320

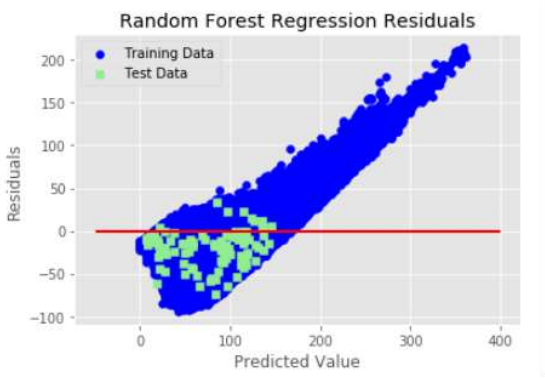
As anticipated during the exploratory data analysis phase, the polynomial regression and random forest algorithm performed better than the other linear models. Random Forest algorithm clearly outperformed other models scoring RMSE of 28.63 i.e. the model is able to predict the failure of airplane engine within the range of +/- 28 cycles.

It should be noted that this model may not be Production Grade and there are lot of avenues of performing the optimization and further data cleansing.

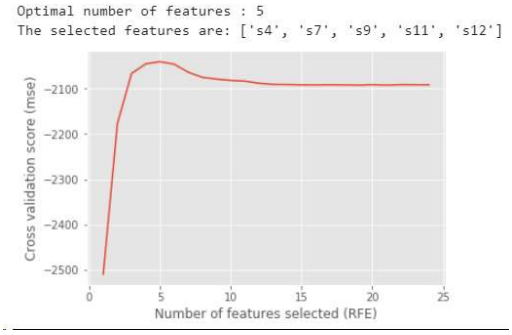
This was also apparent to an extent when we look at the regression residuals. As per the regression residual plots we can see that residuals are randomly spread across the average value of the residuals, and this can be fixed by various optimizations like:

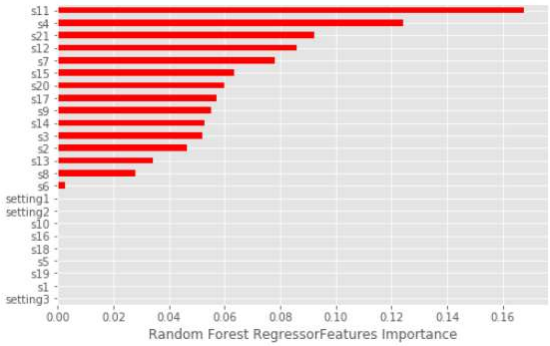
- Removing outliers from the feature data
- Performing hyper parameter tuning for models., etc.





I also examined the key features by looking at the methods like **Recursive Feature Elimination (RFE)** and through the **Feature Importance** returned by Random Forest algorithms





9. Binary Classification – which engine will fail in the current period?

To predict whether an engine will fail in the current period or not, is a binary classification problem. For the experiments we conducted an arbitrary period of 30 cycles was chosen and various binary classification algorithms were run in the test harness.

As was apparent during the exploratory data analysis that we have unbalanced dataset with more observations of class 0 (engine will not fail) as compared to class 1 (engine to fail) so accuracy of the model was not right measure to choose.

I nevertheless calculated accuracy along with various other metrices, but the grid search scoring was done ROC-AUC score to find the best performing model

The list of algorithms chosen for experiments and comparative view of the evaluation metrics is explained as per the list below:

Logistic Regression: is the counterpart of linear regression used in classification problems. In this algorithm the predictions are mapped to 0 and 1 by applying sigmoid function to the predicted probabilities.

Support Vector Machines (SVM): use a mechanism called kernels, which essentially calculate distance between two observations. The SVM algorithm then finds a decision boundary that maximizes the distance between the closes members of two separate classes.

K – Nearest Neighbour: are instance-based algorithms, which means that they save each training observation. They then make predictions for new observation by searching the most similar training observation by polling their values.

Naïve Bayes (NB): is an algorithm around based around conditional probabilities. Essentially, the model is actually a probability table that gets updated through the training data and associated features. To predict a new observation the model simply looks up the class probabilities in the probability table based on its feature value

Following classification performance metrics were calculated and grid search hyper parameter tuning was performed based on AUC-ROC scoring. The table below shows the performance evaluation of the various models tested

	Logistic Regression A	Logistic Regression B	Linear SVC A	Linear SVC B	SVM A	SVM B
Accuracy	0.910000	0.920000	0.880000	0.940000	0.810000	0.910000
Balanced Accuracy	0.820000	0.853333	0.773333	0.933333	0.620000	0.833333
Precision	1.000000	0.947368	0.933333	0.851852	1.000000	0.944444
Recall	0.640000	0.720000	0.560000	0.920000	0.240000	0.680000
F1 Score	0.780488	0.818182	0.700000	0.884615	0.387097	0.790698
ROC AUC	0.980267	0.979200	0.972800	0.977600	0.943467	0.906667

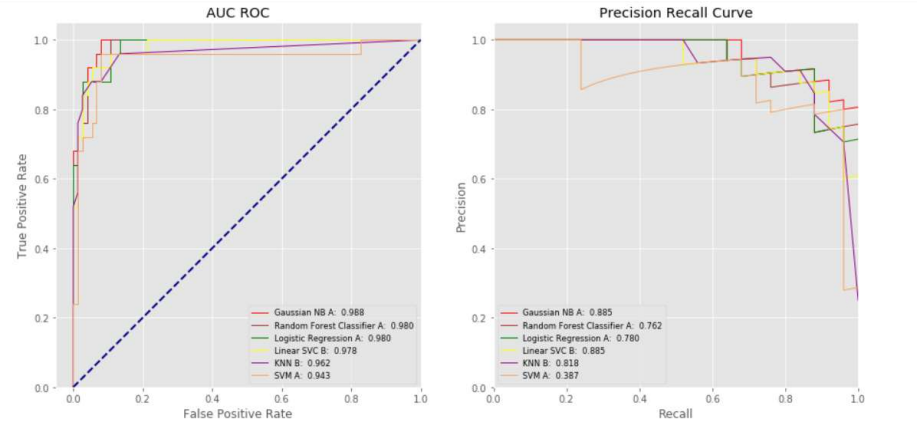
Gaussian NB A	Gaussian NB B	KNN A	KNN B	Random Forest Classifier A	Random Forest Classifier B
0.940000	0.940000	0.910000	0.920000	0.900000	0.900000
0.933333	0.946667	0.833333	0.853333	0.813333	0.813333
0.851852	0.827586	0.944444	0.947368	0.941176	0.941176
0.920000	0.960000	0.680000	0.720000	0.640000	0.640000
0.884615	0.888889	0.790698	0.818182	0.761905	0.761905
0.987733	0.982133	0.935200	0.962400	0.972800	0.974400

```
SVM B                          0.906667
KNN A                          0.935200
SVM A                          0.943467
KNN B                          0.962400
Random Forest Classifier A     0.972800
Linear SVC A                   0.972800
Random Forest Classifier B     0.974400
Linear SVC B                   0.977600
Logistic Regression B          0.979200
Logistic Regression A          0.980267
Gaussian NB B                  0.982133
Gaussian NB A                  0.987733
Name: ROC AUC, dtype: float64
```

Naïve Bayes have performed best for AUC ROC scoring

It was also noticed that feature extraction has improved the performance of most of the machine learning algorithms.

AUC - ROC curves and Precision-Recall curves plotted for some of the best performing models as shown in the chart below:



Expected Benefit Calculation

There should be a way to convert the model's evaluation metrics to expected business benefits.

For this we need to perform the COST BENEFIT analysis by comparing:

- the benefits of correct predictions
- vs the penalties of wrong prediction (i.e. predicting that machine will fail and thus incurring maintenance cost and lost production opportunity)

Such a cost benefit metrics are usually designed by Business Domain Experts / Maintenance Experts. Intuitively speaking it will have weights associated with True Positives, True Negative, False Positives as well as False Negatives and should also consider the constrained operational and maintenance thresholds

According to the book DATA SCIENCE for BUSINESS, a cost benefit matrix can be calculated in line with confusion matrix and then converting the model performance to a single monetary value by multiplying the confusion matrix with the weights for TP, TN, FP and FN.

So, a formula for this could be:

Expected Profit = (+ve probability)* [TPR x Benefit(TP) + FNR x Cost(FN)] + (-ve probability)[TNR x Benefit(TN) + FPR x Cost(FP)]

Let's assume the following numbers:

- True Positive (TP) has benefit of EUR 300 K: the engine's that needs maintenance have been correctly selected by the model
- True Negative (TN) has benefit of EUR 50 K: the engine's that are OK and not selected by model, thus ensuring that we are not doing early maintenance
- False Positive (FP) has cost of EUR -100K: engines that are OK but have been predicted to fail by model, this incurring maintenance cost and loss in production time
- False Negative (FN) has cost of EUR -200K: engines that need maintenance but not selected by model, this will mean that we will have do some urgent maintenance activities which will cost more than the planned maintenance

Highest expected profit/engine calculations for all models were ranked as shown below:

	Profit	Model	Que	Threshold	TP	FP	TN	FN	TPR	FPR	TNR	FNR
0	53.500000	Gaussian NB A	0.31	0.039856	25	0	69	6	1.00	0.080000	0.920000	1.000000
1	52.704225	Gaussian NB B	0.29	0.791340	24	1	70	5	0.96	0.066667	0.933333	0.985915
2	52.473684	Logistic Regression A	0.24	0.211685	22	3	73	2	0.88	0.026667	0.973333	0.960526
3	51.869863	Linear SVC B	0.27	0.036588	23	2	71	4	0.92	0.053333	0.946667	0.972603
4	51.214286	SVM A	0.30	-0.901244	24	1	69	6	0.96	0.080000	0.920000	0.985714
5	51.000000	Logistic Regression B	0.25	0.117968	22	3	72	3	0.88	0.040000	0.960000	0.960000
6	51.000000	Random Forest Classifier A	0.25	0.203848	22	3	72	3	0.88	0.040000	0.960000	0.960000
7	51.000000	Random Forest Classifier A	0.25	0.203848	22	3	72	3	0.88	0.040000	0.960000	0.960000
8	50.097403	KNN B	0.23	0.307692	21	4	73	2	0.84	0.026667	0.973333	0.948052
9	50.097403	SVM B	0.23	-0.540192	21	4	73	2	0.84	0.026667	0.973333	0.948052
10	49.527027	Linear SVC A	0.26	-0.314730	22	3	71	4	0.88	0.053333	0.946667	0.959459
11	46.246753	KNN A	0.23	0.384615	20	5	72	3	0.80	0.040000	0.960000	0.935065

Gaussian Naive Bayes has the best profit per engine (EUR 52 ~ 53 K) if the company has the capacity to maintain 31% of the engines per period (Queue)

The same method could be applied to select the model that gives the best expected profit at specific maintenance capacity level for constrained operations.

	Profit	Model	Que	Threshold	TP	FP	TN	FN	TPR	FPR	TNR	FNR
0	50.097403	KNN B	0.23	0.307692	21	4	73	2	0.84	0.026667	0.973333	0.948052
1	50.097403	SVM B	0.23	-0.540192	21	4	73	2	0.84	0.026667	0.973333	0.948052
2	46.246753	KNN A	0.23	0.384615	20	5	72	3	0.80	0.040000	0.960000	0.935065
3	52.473684	Logistic Regression A	0.24	0.211685	22	3	73	2	0.88	0.026667	0.973333	0.960526
4	51.000000	Logistic Regression B	0.25	0.117968	22	3	72	3	0.88	0.040000	0.960000	0.960000
5	49.527027	Linear SVC A	0.26	-0.314730	22	3	71	4	0.88	0.053333	0.946667	0.959459
6	51.869863	Linear SVC B	0.27	0.036588	23	2	71	4	0.92	0.053333	0.946667	0.972603
7	52.704225	Gaussian NB B	0.29	0.791340	24	1	70	5	0.96	0.066667	0.933333	0.985915
8	51.214286	SVM A	0.30	-0.901244	24	1	69	6	0.96	0.080000	0.920000	0.985714
9	53.500000	Gaussian NB A	0.31	0.039856	25	0	69	6	1.00	0.080000	0.920000	1.000000
10	49.724638	Random Forest Classifier A	0.31	0.128068	24	1	68	7	0.96	0.093333	0.906667	0.985507
11	49.724638	Random Forest Classifier A	0.31	0.128068	24	1	68	7	0.96	0.093333	0.906667	0.985507

For constrained operations:

- Working at capacity of 27% -> Linear SVC gives Best Profit of 51.86K EUR per engine
- Working at capacity of 23% -> KNN gives Best Profit of 50.09K EUR per engine

10. Multiclass Classification – Improved PdM Planning

To answer this question, engines remaining cycles or TTF were segmented into bands of cycles, for example period

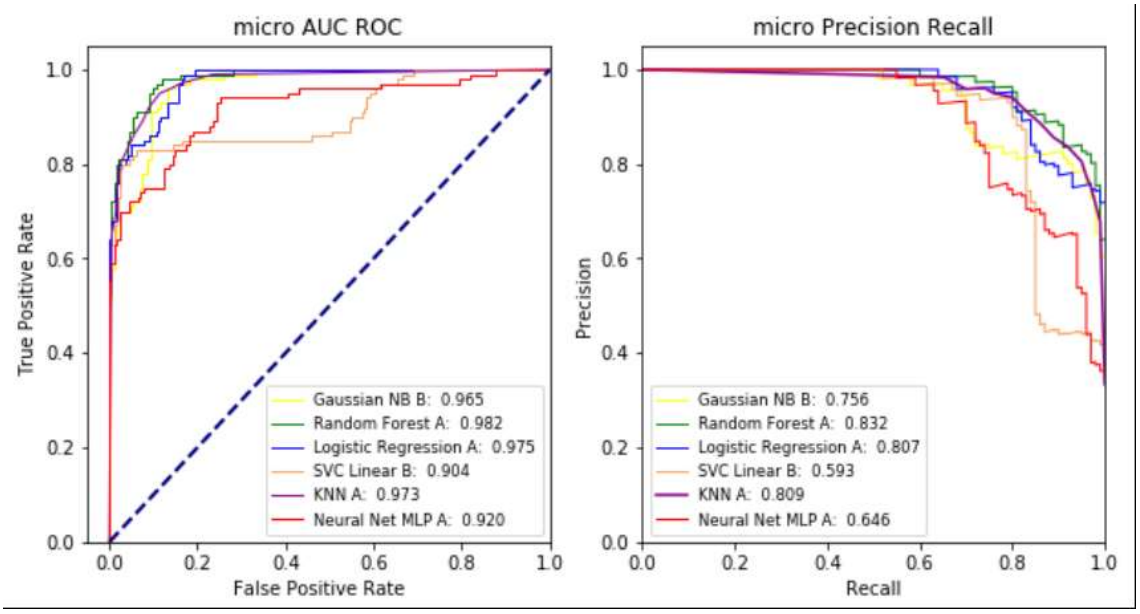
- Period 0: 0 –15 cycles,
- period 1: 16 –30 cycles, and
- period 2: 30+ cycles.

Then, multiclass classification algorithms were used to predict the period which an engine will fail in.

Machine learning algorithms tried included Logistic Regression, Decision Trees, Support Vector Machines, K Nearest Neighbors, Gaussian Naive Bayes, Random Forests, and Neural Networks MLP.

For model evaluation, micro and macro averages of AUC ROC, Recall, Precision, and F1 were calculated in addition to Accuracy. Values of these metrics on the test dataset are shown below:

	Accuracy	Macro F1	Micro F1	Macro Precision	Micro Precision	Macro Recall	Micro Recall	Macro ROC AUC	Micro ROC AUC
Logistic Regression B	0.83	0.597157	0.864583	0.597501	0.902174	0.600000	0.83	0.948393	0.972900
Logistic Regression A	0.82	0.596941	0.863158	0.600823	0.911111	0.595556	0.82	0.945572	0.974550
SVC Linear B	0.65	0.430120	0.729614	0.353040	0.639098	0.666667	0.85	0.934898	0.903950
SVC Linear A	0.62	0.299517	0.760736	0.328042	0.984127	0.275556	0.62	0.942490	0.950900
Gaussian NB B	0.76	0.754885	0.854545	0.658811	0.783333	0.937778	0.94	0.950334	0.965350
Gaussian NB A	0.74	0.754954	0.849315	0.664502	0.781513	0.933333	0.93	0.944855	0.950325
KNN B	0.83	0.641710	0.855670	0.800813	0.882979	0.595556	0.83	0.904947	0.954825
KNN A	0.86	0.720426	0.873096	0.804527	0.886598	0.673333	0.86	0.948197	0.973225
Random Forest B	0.82	0.603520	0.858639	0.897119	0.901099	0.584444	0.82	0.966697	0.979050
Random Forest A	0.87	0.755671	0.887755	0.878601	0.906250	0.706667	0.87	0.970760	0.982200
Neural Net MLP B	0.75	0.285714	0.750000	0.250000	0.750000	0.333333	0.75	0.767072	0.880800
Neural Net MLP A	0.75	0.285714	0.750000	0.250000	0.750000	0.333333	0.75	0.854053	0.920400



The graph shows that Naive Bayes and Random Forest are top two performing models in both AUC ROC and Precision-Recall curves.

11. Summary and Next Steps

The project has explored the mechanisms to answer three essential questions in the predictive maintenance., i.e.,

- When an engine will fail
- Whether an engine will fail in current period or not
- How to better plan maintenance based on prediction across multiple periods

By applying the machine learning algorithms for regression, binary classification and multiclass classification on historical data from sensor readings, the project was able to demonstrate how this journey may look like

In this project we have not focused on creating a Production grade model, but rather on what are the various possibilities. So, as a next step we can optimize the models in all the three groups by doing following activities:

- Perform data cleaning based on various approaches discussed during exploratory data analysis section (outlier detection and removal, scaling of the data, resampling and testing with various window sizes, tuning model parameters etc.)

And the final optimized model can be deployed for online access and for live testing with the actual machines