PREDICTIVE MAINTENANCE

5/18/2017

Data Science Project Final Report

Author: Sami Mustafa

Mentor: Alex Chao

The main promise of predictive maintenance is to reduce the cost of time-based maintenance, and to prevent unexpected equipment failures. By knowing which equipment needs maintenance, repair work can be better planned. This report describes the work and the results of applying data science techniques to aircraft engine sensors measurements to predict in-service engine failure.

This project has been done in fulfillment to the First Capstone Project requirement of Springboard DS Bootcamp

Predictive Maintenance

DATA SCIENCE PROJECT FINAL REPORT

Introduction

Airlines are interested in predicting engines failures in advance to enhance operations and reduce flight delays. Observing engine's health and condition through sensors and telemetry data is assumed to facilitate this type of maintenance by predicting Time-To-Failure (TTF) of in-service engine. Consequently, maintenance work could be planned according to TTF predictions instead of/to complement costly time-based preventive maintenance.

This report summarizes the work done and the results of using machine learning and data science techniques to acquire, clean, and analyze aircraft engines sensors measurements, and the predictions results generated by different machine learning algorithms.

This work has been done as part of the Springboard Data Science Career Track training to fulfill the requirements of the First Capstone Project.

Approach

By exploring aircraft engine's sensor values over time, machine learning algorithm can learn the relationship between sensor values and changes in sensor values to the historical failures in order to predict failures in the future.

Supervised machine learning algorithms will be used to make the following predictions:

- Use of Regression algorithms to predict engine TTF
- Use of Binary Classification algorithms to predict if the engine will fail in this period
- Use of Multiclass Classification algorithms to predict the period an engine will fail

Development Environment

- Jupyter Notebook running on Python 3.5.2
- Pandas '0.19.2' and Numpy '1.11.2' for data wrangling and analysis
- Matplotlib '1.5.3' and Seaborn '0.7.1' for visualization
- Scikit-learn '0.18.1' for machine learning algorithms

Project repo on GitHub: https://github.com/samimust/predictive-maintenance

Data Acquisition

Files contain simulate aircraft engine run-to-failure events, operational settings, and sensors measurements were provided by Microsoft Cortana Intelligence.

- Training data file contains aircraft engines' run-to-failure data. 20,000+ cycle records for 100 engines.
- Test data file contains aircraft engines' operating data without failure events recorded. Remaining cycles for each engine is provided in a separate Ground truth data file.
- Ground truth data file contains the true remaining cycles for each engine in the test data.

Sample training data:

id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	...	s15	s16	s17	s18	s19	s20	s21
1	1	-0.0007	-0.0004	100	518.67	641.82	1589.7	1400.6	14.62	21.61	554.36	2388.06		8.4195	0.03	392	2388	100	39.06	23.419
1	2	0.0019	-0.0003	100	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04		8.4318	0.03	392	2388	100	39	23.4236
1	3	-0.0043	0.0003	100	518.67	642.35	1587.99	1404.2	14.62	21.61	554.26	2388.08		8.4178	0.03	390	2388	100	38.95	23.3442
100	198	0.0004	0	100	518.67	643.42	1602.46	1428.18	14.62	21.61	550.94	2388.24		8.5646	0.03	398	2388	100	38.44	22.9333
100	199	-0.0011	0.0003	100	518.67	643.23	1605.26	1426.53	14.62	21.61	550.68	2388.25		8.5389	0.03	395	2388	100	38.29	23.064
100	200	-0.0032	-0.0005	100	518.67	643.85	1600.38	1432.14	14.62	21.61	550.79	2388.26		8.5036	0.03	396	2388	100	38.37	23.0522

- **id**: is the engine ID, ranging from 1 to 100
- **cycle**: per engine sequence, starts from1 to the cycle number where failure had happened
- **setting1** to **setting3**: engine operational settings
- **s1** to **s21**: sensors measurements

Data Wrangling

All features columns are numeric with no missing values.

Regression and classification labels for training data were created as follows:

- Regression: Time-To-Failure TTF (no. of remaining cycle before failure) for each cycle/engine is the number of cycles between that cycle and the last cycle of the same engine.
- Binary Classification: if the remaining cycles is less than specific number of cycles (e.g. period = 30), the engine will fail in this period, otherwise the engine is fine.
- Multiclass Classification: by segmenting the TTF into cycle bands (e.g. periods: 0-15, 16-30, 30+), we could identify in which period will the engine fail.

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

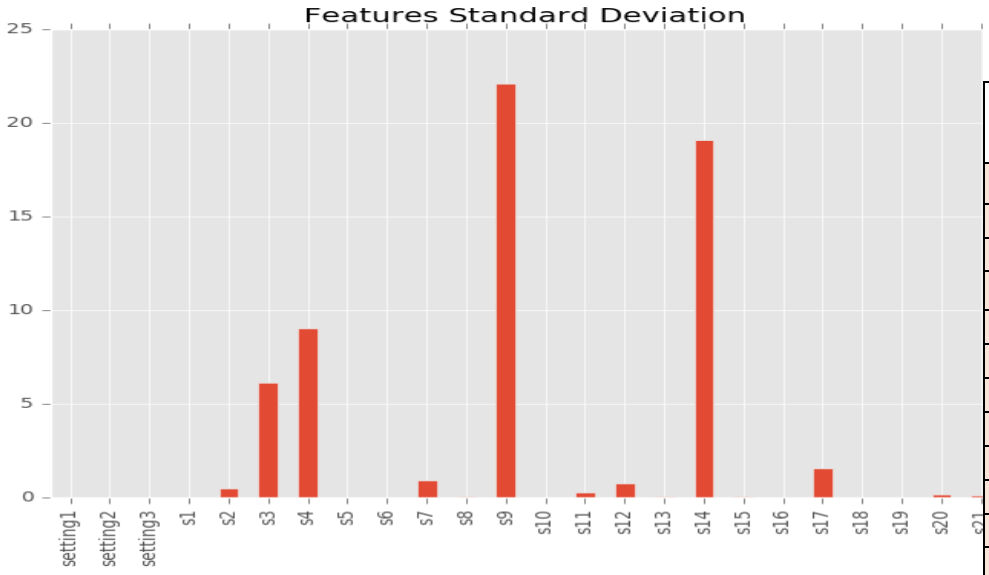
Feature extraction is also applied to the training and test data by introducing additional two columns for each of the 21 sensor columns: rolling mean and rolling standard deviation. This smoothing to the sensors measurements over time would improve the performance of some machine learning algorithms.

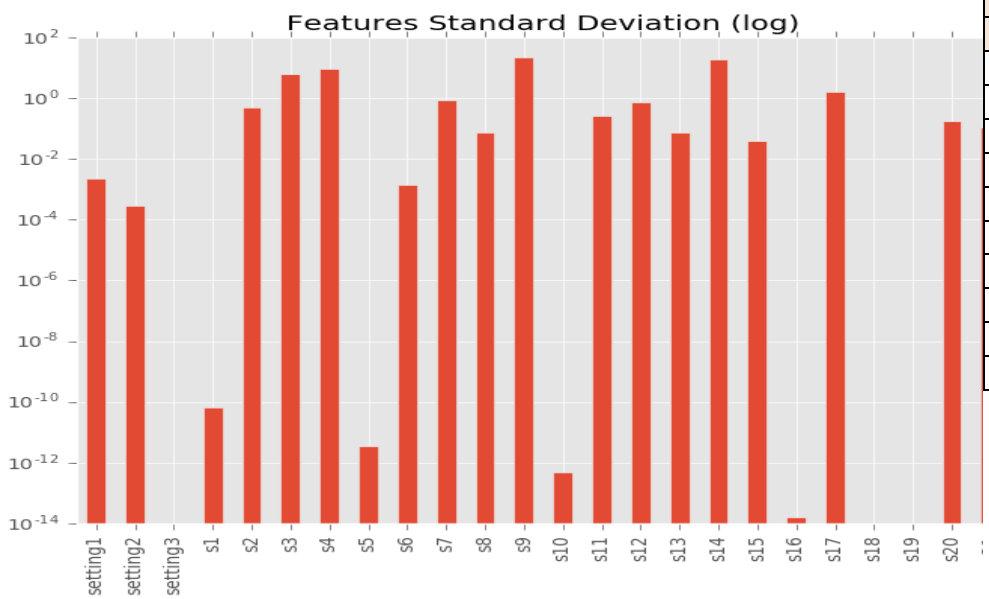
The final training and test data were saved to csv files ready for next steps of analysis and modeling.

Data Wrangling Notebook on GitHub

Exploratory Data Analysis

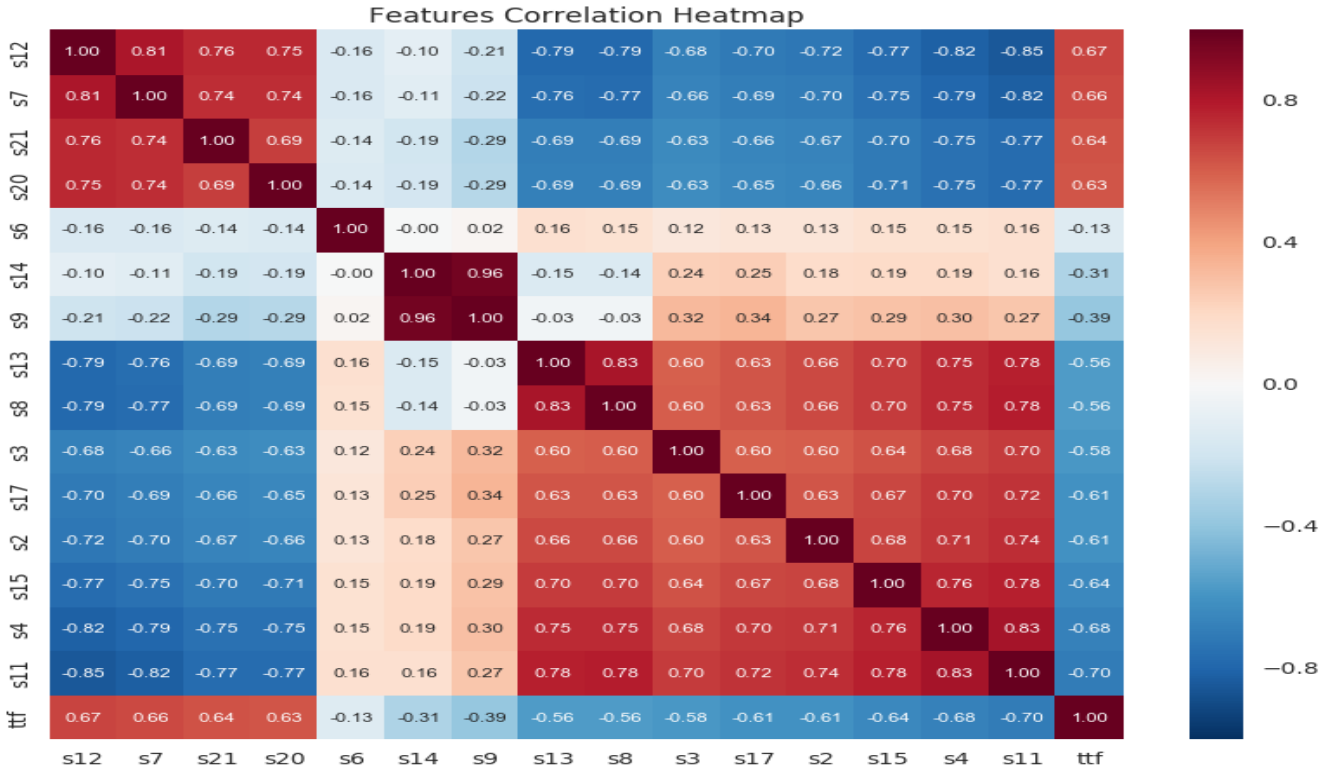
Feature variability, distribution, and correlation were examined to uncover underlying structure and extract important variables.



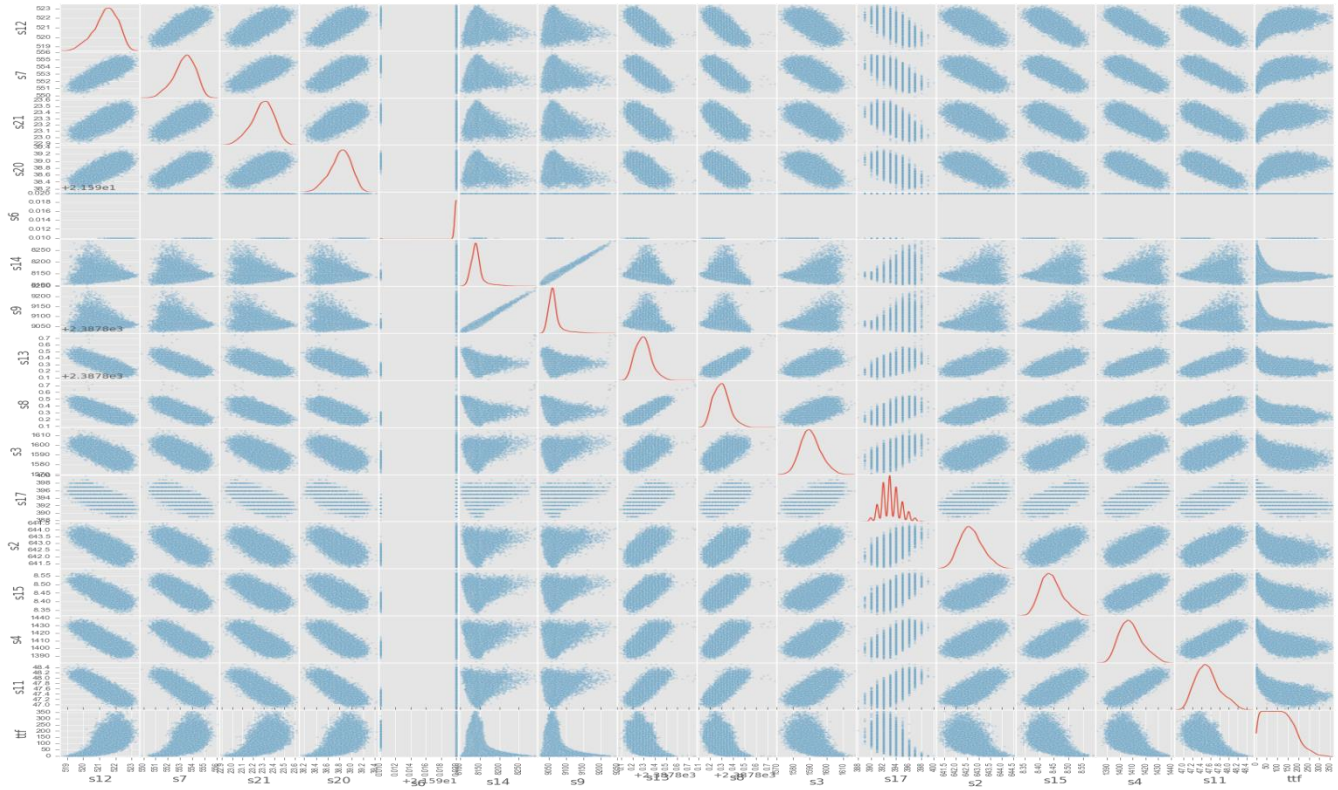


Feature	Standard Deviation
s9	22.0829
s14	19.0762
s4	9.0006
s3	6.1312
s17	1.5488
s7	0.8851
s12	0.7376
s2	0.5001
s11	0.2671
s20	0.1807
s21	0.1083
s13	0.0719
s8	0.0710
s15	0.0375
setting1	0.0022
s6	0.0014
setting2	0.0003
s1	0.0000
s5	0.0000
s10	0.0000
s16	0.0000
s18	-
s19	-
setting3	-

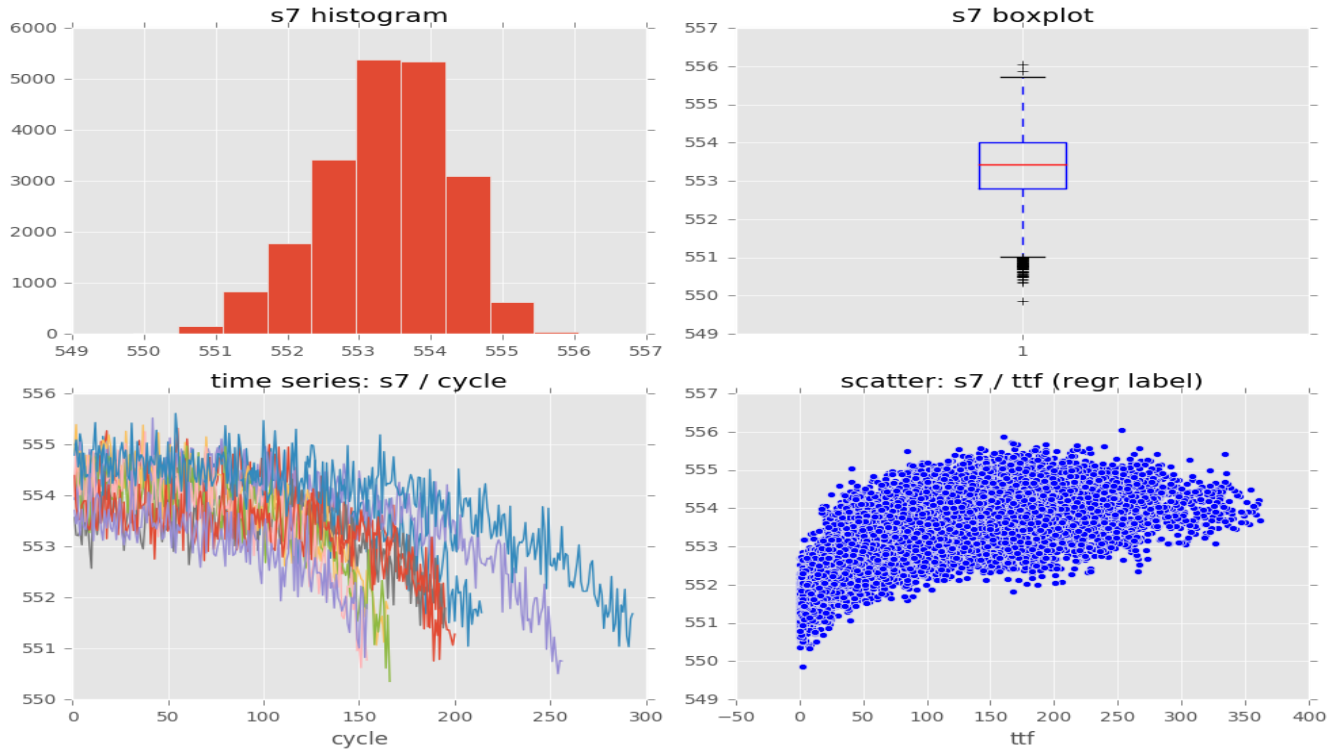
Features with high variability were checked for correlation with other features and regression label (TTF):



Scatter matrix was also used to check the distribution and correlation of features:



A number of EDA charts were also used to have more insights on each feature individually, e.g. **S7**:



- There is a very high correlation (> 0.8) between some features e.g.:
  (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, part of these features will be target for elimination in feature selection during the modeling phase.
- Most features have nonlinear relation with the TTF, hence adding their polynomial transforms may enhance models performance.
- Most features exhibit normal distribution which is likely improves models performance.

Exploratory Data Analysis Notebook on GitHub

Predicting Engine's Time-To-Failure (TTF)

Both linear and non-linear regression models were tried to predict engine's TTF. The following machine learning algorithms were tried and their performance metrics were calculated and evaluated: Linear Regression, LASSO Regression, Ridge Regression, Decision Tree Regression, Polynomial Regression, and Random Forests Regression. A short definition of these algorithms:

Linear Regression: statistical method of analysis that estimates the relationship between one or more independent variables and a dependent variable; the method estimates the relationship by minimizing the sum of the squares in the difference between the observed and predicted values of the dependent variable configured as a straight line or hyperplane.

LASSO: (Least Absolute Shrinkage and Selection Operator) is a regularized version of least squares regression. It minimizes the sum of squared errors while also penalizing the L1 norm (sum of absolute values) of the coefficients.

Ridge Regression: Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.

Polynomial Regression: is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an nth degree polynomial in x

Decision Trees: learn in a hierarchical fashion by repeatedly splitting the dataset into separate branches that maximize the information gain of each split. In regression tree, the value obtained by terminal nodes in the training data is the mean response of observation falling in that region, whereas in classification tree, the value (class) obtained by terminal node in the training data is the mode of observations falling in that region.

Random Forests: are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

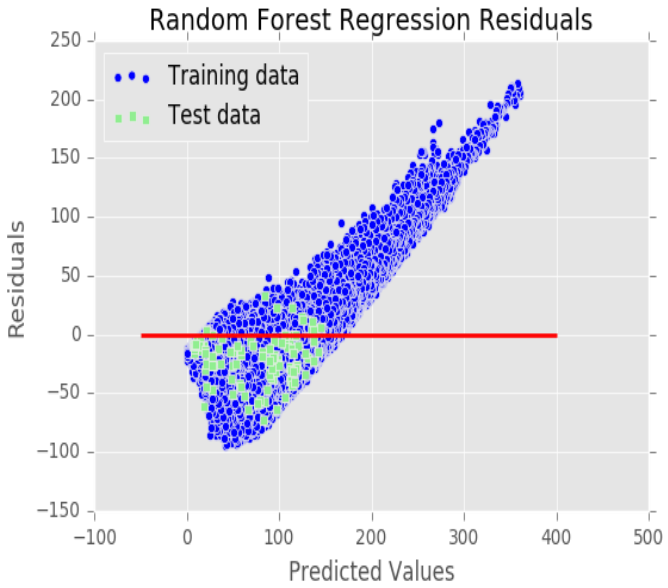
Key regression evaluation metrics calculated were Root Mean Squared Error (RMSE), R-squared (R2), Mean Absolute Error, and Explained Variance. The results on test dataset listed below:

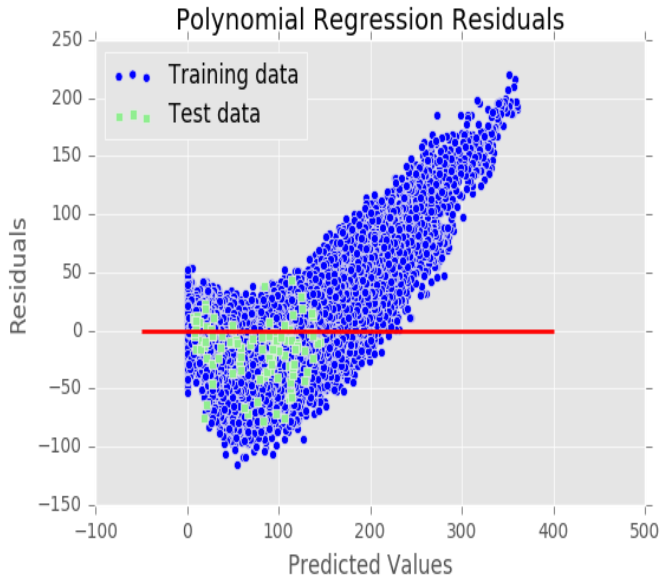
	Linear	LASSO	Ridge	Decision Tree	Polynomial	Random Forests
Root Mean Squared Error	32.04	31.97	31.97	32.10	29.68	28.63
Mean Absolute Error	25.59	25.55	25.54	24.32	22.38	23.17
R-Squared (R^2)	0.41	0.41	0.41	0.40	0.49	0.53
Explained Variance	0.67	0.67	0.67	0.63	0.65	0.77

In accordance with our analysis in the data exploratory phase, non-linear regression models like Polynomial and Random Forest performed better than linear models like OLS, LASSO and Ridge regression. Random Forest clearly outperformed other models scoring RMSE of 28.63 cycles, i.e. the model predicts TTF within average error range of ±28.63 cycles. First 10 predictions of Random Forest model on test dataset were:

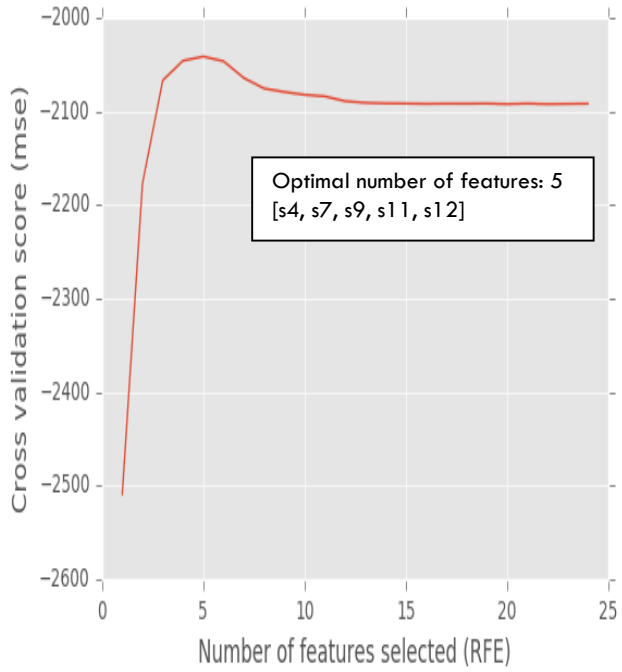
	0	1	2	3	4	5	6	7	8	9	10
Actual	112.00	98.00	69.00	82.00	91.00	93.00	91.00	95.00	111.00	96.00	97.00
Prediction	151.58	119.27	74.42	96.47	112.59	130.28	128.11	100.69	116.12	127.37	74.37
Difference	-39.58	-21.27	-5.42	-14.47	-21.59	-37.28	-37.11	-5.69	-5.12	-31.37	22.63

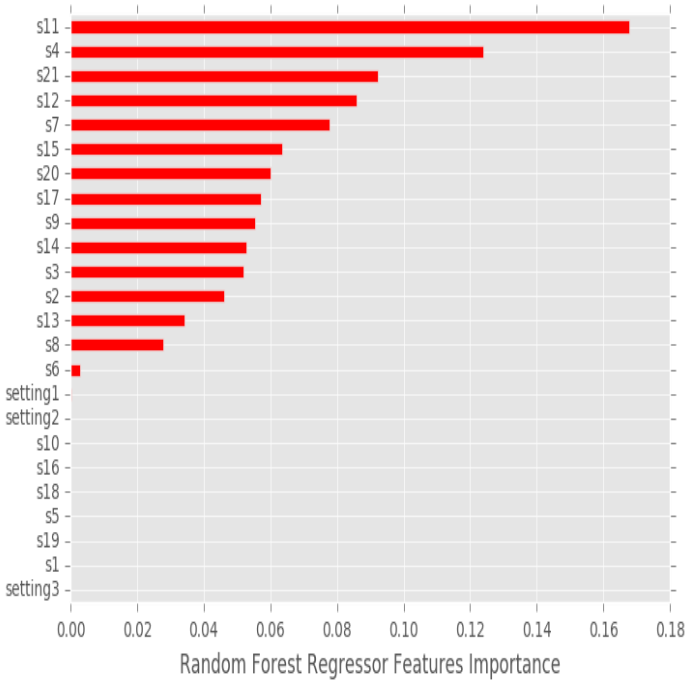
As per the regression residuals plot shown below, residuals were not randomly spread across the average value of the residuals. This could be improved by many methods including fixing the data (e.g. outliers), model parameters tuning, or trying other ML algorithms.





Key features were also examined through feature selection methods like Recursive Feature Elimination (RFE) and through parameters returned by models like Random Forest *feature importance* as shown below:





Model Selection - Linear Regression Notebook on GitHub

Which Engines will fail in the Current Period?

Multiple binary classification algorithms were used to predict which engines will fail in the current period, e.g. remaining cycles or TTF in the range 0-30 cycles. Evaluated algorithms include Logistic Regression, Decision Trees, Support Vector Machines, K Nearest Neighbors, Naive Bayes, and Random Forests. A short definition of these algorithms:

Logistic Regression: is the classification counterpart to linear regression. Predictions are mapped to be between 0 and 1 through the logistic function, which means that predictions can be interpreted as class 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 closest members of separate classes.

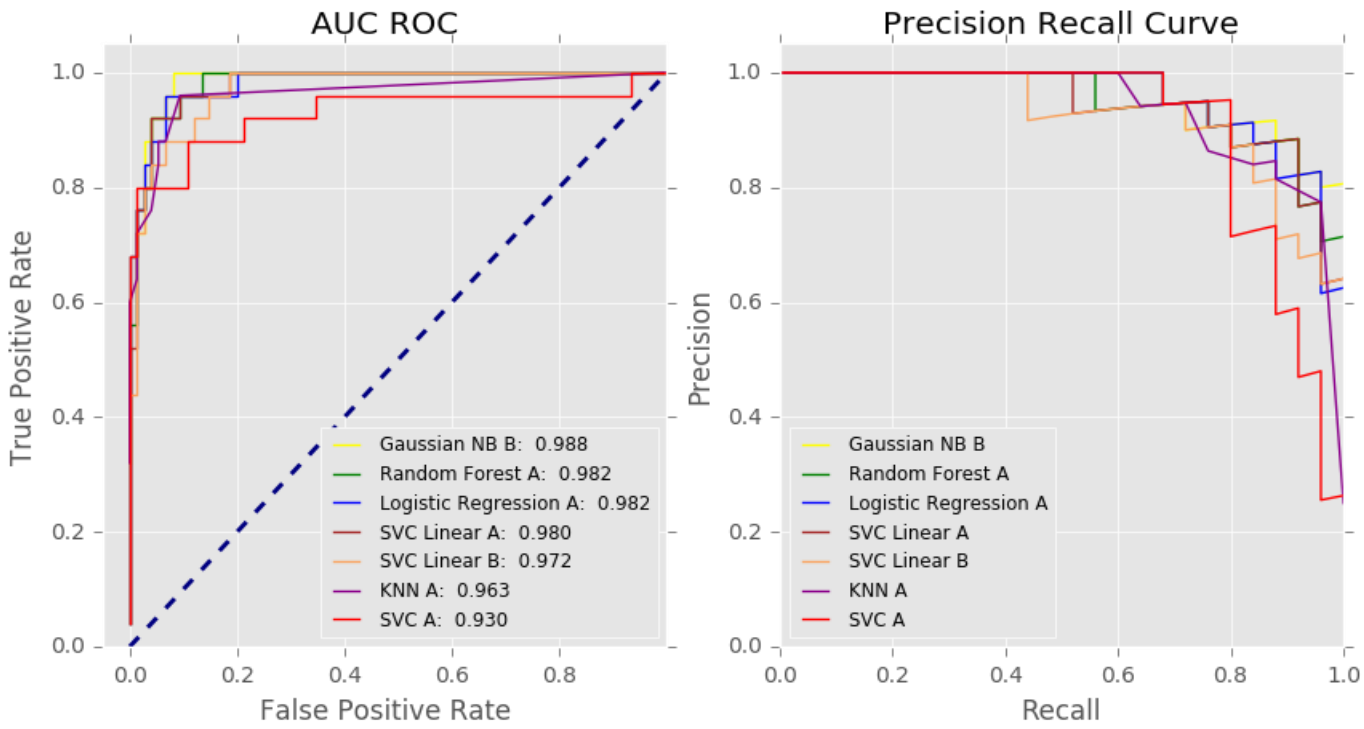
Nearest Neighbors: are instance-based algorithms, which mean that they save each training observation. They then make predictions for new observations by searching for the most similar training observations and pooling their values.

Naive Bayes (NB): is a very simple algorithm based around conditional probability and counting. Essentially, the model is actually a probability table that gets updated through the training data. To predict a new observation, the model simply look-up the class probabilities in the probability table based on its feature values.

Key binary classification performance metrics calculated include Area under Receiver Operating Characteristics Curve (AUC ROC), Recall, Precision, F1 Score, and Accuracy. AUC ROC was the score used in Grid Search hyper-parameters tuning. As shown below, performance of various models was evaluated on the test dataset, where **B** stands for original features set (**B**efore features extraction), and **A** stand for modified feature set (**A**fter feature extraction).

	Logistic Reg. B	Logistic Reg. A	Decision Tree B	Decision Tree A	Random Forest B	Random Forest A	SVC B	SVC A	Linear SVC B	Linear SVC A	KNN B	KNN A	Gaussian NB B	Gaussian NB A
AUC ROC	0.9803	0.9819	0.9451	0.9629	0.9803	0.9824	0.8917	0.9301	0.9717	0.9797	0.9352	0.9635	0.9877	0.9805
Precision	0.9333	1.0000	0.9333	0.9474	0.9444	0.9444	0.9444	0.9474	1.0000	0.4310	0.9444	0.9474	0.8276	0.8276
Recall	0.5600	0.6800	0.5600	0.7200	0.6800	0.6800	0.6800	0.7200	0.4000	1.0000	0.6800	0.7200	0.9600	0.9600
F1 Score	0.7000	0.8095	0.7000	0.8182	0.7907	0.7907	0.7907	0.8182	0.5714	0.6024	0.7907	0.8182	0.8889	0.8889
Accuracy	0.8800	0.9200	0.8800	0.9200	0.9100	0.9100	0.9100	0.9200	0.8500	0.6700	0.9100	0.9200	0.9400	0.9400

Naïve Bayes and Random Forests scored best AUC ROC. It also noticed that feature extraction has improved most models performance metrics. AUC for ROC and Precision-Recall curves were plotted for best models as shown below:



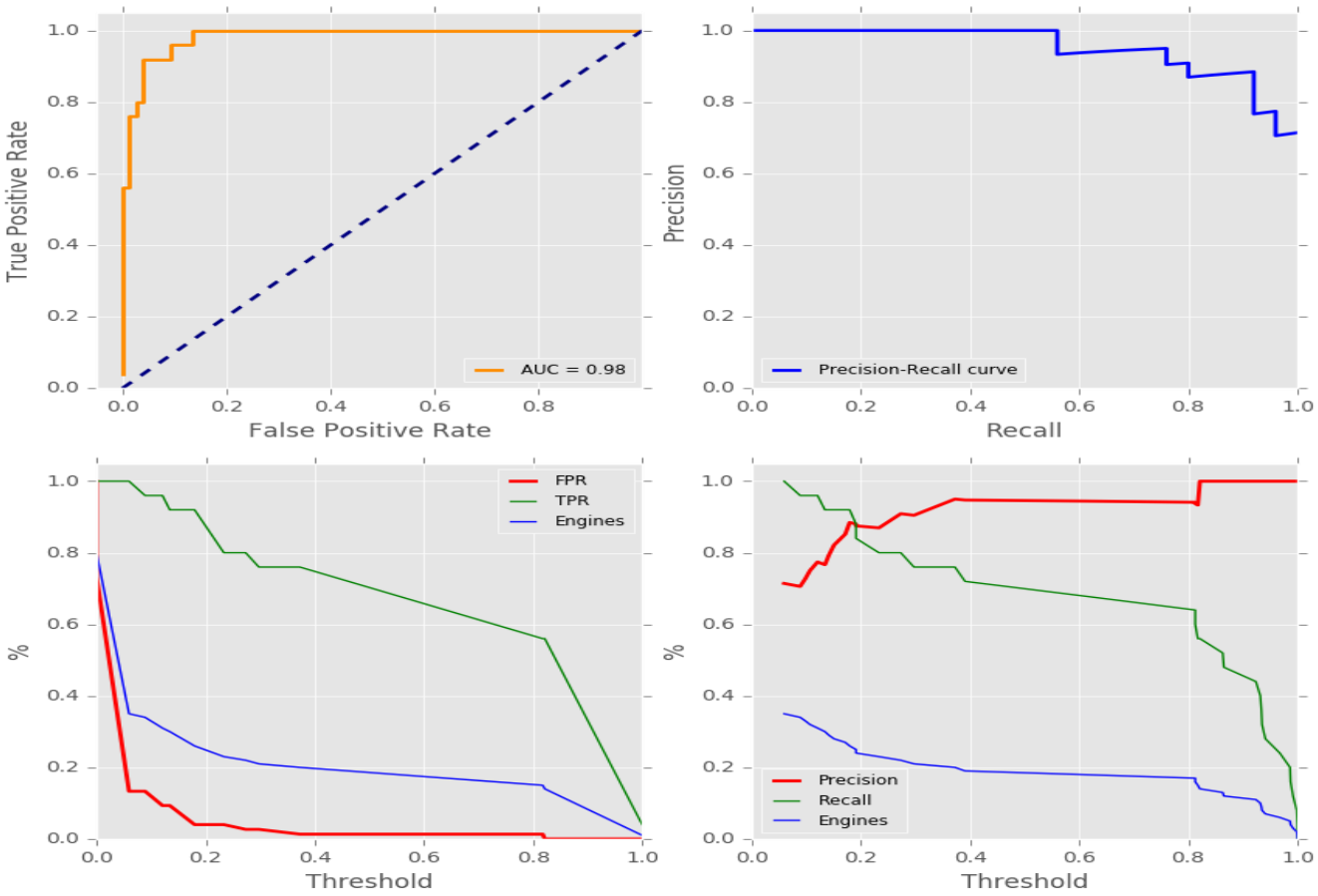
For each model, a number of evaluation charts and metrics were developed as shown below for Random Forests:

```
Confusion Matrix:
[[74  1]
 [ 8 17]]
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.99      0.94        75
           1       0.94      0.68      0.79        25
```

```
avg / total       0.91      0.91      0.90       100
```



Engines in the above charts represent the queue or number of engines to be maintain per period, i.e. maintenance capacity. With the ability to threshold at different level, maximum business gain could be achieved based on business capacity, Recall, and Precision targets. For example, the above model selected the threshold of 0.6 to give %68 Recall, %94.4 Precision for 18 engines. If there is no issue with maintenance capacity, we could select to threshold the model at 0.4, resulting %72 Recall, %94.7 Precision for 19 engines, achieving better results.

	Threshold	Precision	Recall	Engines
..				
14	0.297494	0.904762	0.76	0.21
15	0.372589	0.950000	0.76	0.20
16	0.391062	0.947368	0.72	0.19
17	0.603777	0.944444	0.68	0.18
18	0.812644	0.941176	0.64	0.17
19	0.812659	0.937500	0.60	0.16
..				

Expected Profit

According to the book Data Science for Business, different classification models could be compared using the Expected Value calculation. This is achieved by constructing a cost-benefit matrix in line with the model confusion matrix, and then converting model performance to a single monetary value by multiplying confusion matrix into the cost-benefit matrix using the formula:

Expected Profit = Probability(+ve) x [TPR x benefit(TP) + FNR x cost(FN)] + Probability(-ve) x [TNR x benefit(TN) + FPR x cost(FP)]

Cost-benefit matrix should be provided by business domain experts. For this project, the following values were assumed:

- True Positive (TP): engines need maintenance and selected by the model, has benefit of $300K
- True Negative (TN): engines that are fine and not selected by the model, has benefit of $0K
- False Positive (FP): engines that are fine but selected by the model, has cost of $-100K
- False Negative (FN): engines need maintenance but not selected by the model, has cost of $-200K

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

Rank	Exp. Profit	Model	Engines	Threshold	TP	FP	TN	FN	TPR	FPR	TNR	FNR
0	19.00	Gaussian NB B	0.31	0.09	25	0	69	6	1.00	0.08	0.92	1.00
1	18.69	Logistic Regression B	0.28	0.11	24	1	71	4	0.96	0.05	0.95	0.99
2	18.69	Gaussian NB A	0.28	0.97	24	1	71	4	0.96	0.05	0.95	0.99
3	17.70	Logistic Regression A	0.29	0.06	24	1	70	5	0.96	0.07	0.93	0.99
4	17.35	Random Forest A	0.26	0.18	23	2	72	3	0.92	0.04	0.96	0.97
5	17.35	SVC Linear A	0.26	0.78	23	2	72	3	0.92	0.04	0.96	0.97
6	17.00	Random Forest B	0.33	0.10	25	0	67	8	1.00	0.11	0.89	1.00
7	15.72	KNN A	0.31	0.08	24	1	68	7	0.96	0.09	0.91	0.99
8	13.05	SVC Linear B	0.27	(0.59)	22	3	70	5	0.88	0.07	0.93	0.96
9	12.16	SVC A	0.21	(0.23)	20	5	74	1	0.80	0.01	0.99	0.94
10	12.08	SVC B	0.28	(0.94)	22	3	69	6	0.88	0.08	0.92	0.96
11	10.70	KNN B	0.26	0.31	21	4	70	5	0.84	0.07	0.93	0.95
12	10.14	Decision Tree A	0.30	0.18	22	3	67	8	0.88	0.11	0.89	0.96
13	7.82	Decision Tree B	0.29	0.08	21	4	67	8	0.84	0.11	0.89	0.94

So, if there is a maintenance capacity to serve %31 engines per period, Naïve Bayes predictions will produce expected profit of $19K per engine.

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

Model Selection – Binary Classification Notebook on GitHub

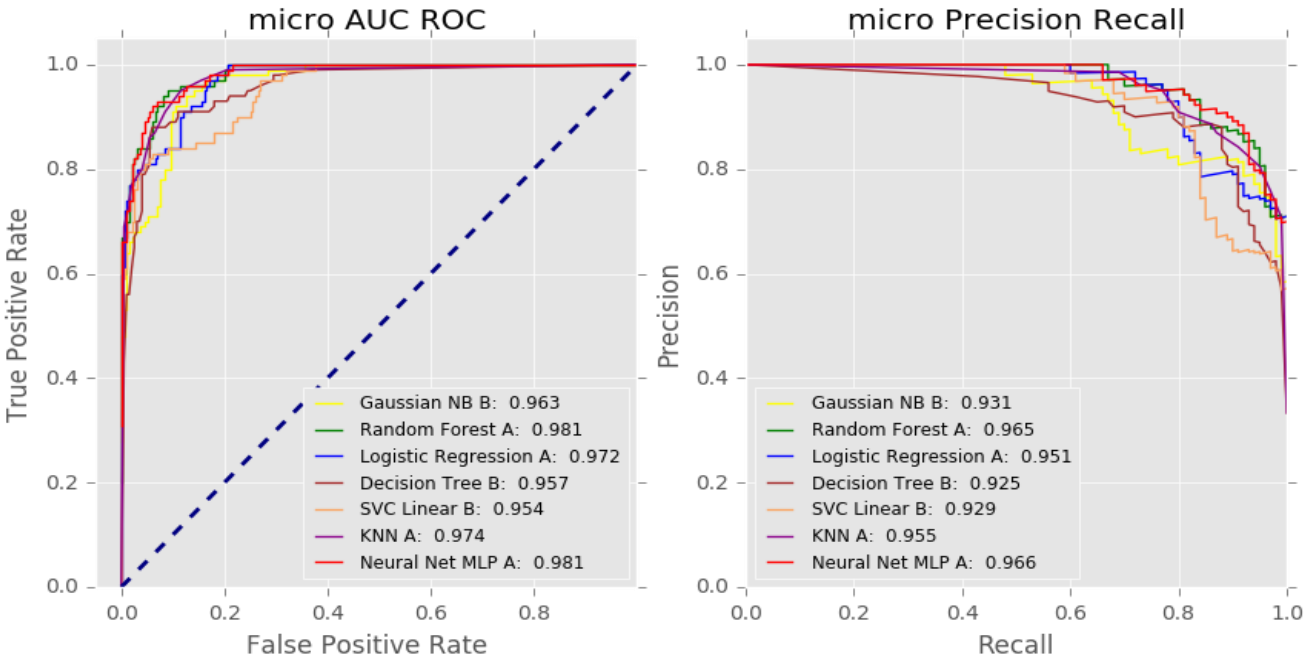
How could Maintenance be better Planned?

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

	micro ROC AUC	macro ROC AUC	micro Recall	macro Recall	micro Precision	macro Precision	micro F1	macro F1	Accuracy
Logistic Regression B	0.9705	0.9452	0.8100	0.5622	0.8804	0.5564	0.8438	0.5579	0.8100
Logistic Regression A	0.9718	0.9415	0.8100	0.5333	0.9000	0.5869	0.8526	0.5517	0.8100
Decision Tree B	0.9566	0.9056	0.8400	0.6689	0.8842	0.8190	0.8615	0.6841	0.8400
Decision Tree A	0.9736	0.9499	0.8400	0.6511	0.8750	0.8521	0.8571	0.6079	0.8400
Random Forest B	0.9785	0.9643	0.8200	0.5733	0.8913	0.7767	0.8542	0.6125	0.8200
Random Forest A	0.9806	0.9677	0.8500	0.6622	0.8854	0.8008	0.8673	0.7058	0.8500
SVC Linear B	0.9537	0.9347	0.6800	0.4178	0.9714	0.5949	0.8000	0.4825	0.6800
SVC Linear A	0.9172	0.9433	0.9200	0.7333	0.4792	0.6611	0.6301	0.5011	0.0200
KNN B	0.9548	0.9049	0.8300	0.5956	0.8830	0.8008	0.8557	0.6417	0.8300
KNN A	0.9735	0.9499	0.8600	0.6844	0.8866	0.7934	0.8731	0.7099	0.8600
Gaussian NB B	0.9627	0.9503	0.9500	0.9778	0.7724	0.6556	0.8520	0.7579	0.7400
Gaussian NB A	0.9429	0.9448	0.9300	0.9333	0.7815	0.6645	0.8493	0.7550	0.7400
Neural Net MLP B	0.9833	0.9706	0.8500	0.6689	0.9043	0.8736	0.8763	0.7320	0.8500
Neural Net MLP A	0.9813	0.9709	0.8900	0.7622	0.9082	0.8902	0.8990	0.7981	0.8800



Neural Net Multi-layer Perceptron classifier clearly outperformed other models in all metrics, with the Random Forests classifier scoring in the second place.

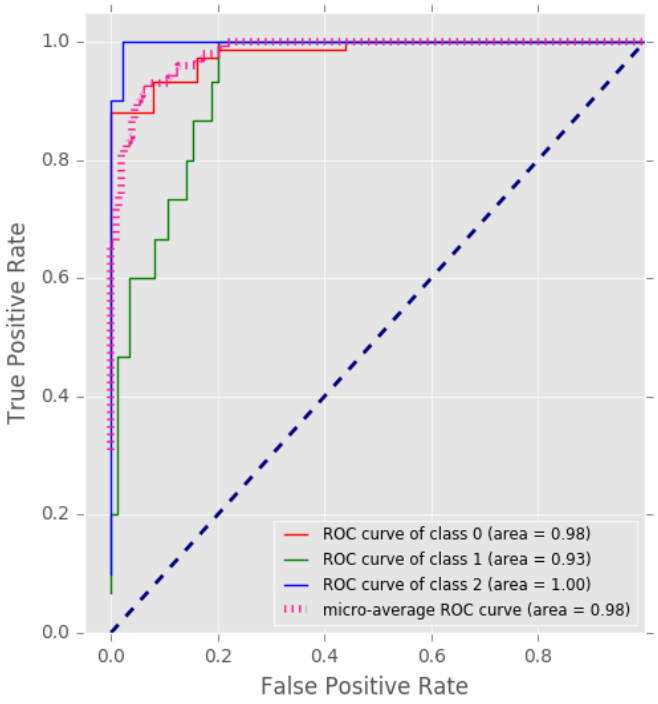
Detailed performance results of the Neural Net MLP classifier are shown below:

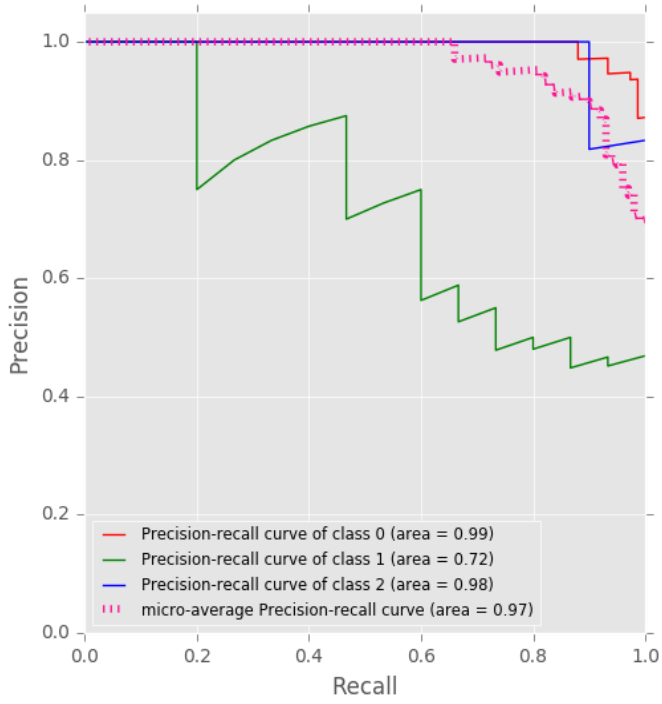
```
Confusion Matrix:
[[75  0  0]
 [ 9  5  1]
 [ 0  1  9]]
```

```
Classification Report:
             precision    recall  f1-score   support

          0       0.89      1.00      0.94        75
          1       0.83      0.33      0.48        15
          2       0.90      0.90      0.90        10

avg / total       0.88      0.89      0.87       100
```





Model Selection – Multiclass Classification Notebook on GitHub

Summary and Next Steps

The project tried to answer three essential questions in predictive maintenance: When an engine will fail? Which engines will fail in this period? How better could maintenance be scheduled?

By applying machine leaning regression, binary classification, and multiclass classification algorithms respectively, to historical data of engines sensors, the project was able to provide some suggestions responding to the problem.

Since predicting TTF is critical to all kinds of modeling performed in this project, more work is required to enhance regression performance. This could be by fixing data (outliers, resampling etc.), trying other models, or tuning models parameters.

Features selection and dimensionality reduction techniques should also be utilized to enhance models performance and speed. Neural Net and SVM with RBF kernel required extensive computation and time.

Finally, the selected model in each category should be deployed for online accessibility.