

# USING RANK PERMUTATION FOR AIRCRAFT ENGINE FAULT DIAGNOSTICS

Xiao Hu, Neil Eklund, Kai Goebel

GE Global Research  
One Research Circle, Niskayuna, NY 12309  
{hux, eklund, goebelk }@research.ge.com

## **Abstract:**

Accurate and timely detection and diagnosis of aircraft engine fault is critical to the normal operation of engine/airplane and to maintain them in a healthy state. In engine fault diagnostics, engine gas path measurements, such as exhaust gas temperature (EGT), fuel flow (WF) and core speed (N2), etc. are frequently used. Some diagnostics models employ trend shift detection for these measurements, a process used to identify when the engine performance data start to change away from their nominal level. Another method to recognize faults is to employ customized on-line built-in tests that inform about abnormal conditions by producing an error log. This paper presents a trend shift detection approach that reasons over features extracted both from sensor measurements as well as from error logs. Specifically, we use the rank permutation transformation (RPT) to produce these condition-features. A suite of diagnostic models is then built up to integrate selected features to make a diagnostic decision. The models are compared and results from real engine data are presented.

**Keyword:** diagnostics; error log messages; leave-one-out; neural networks; random forest; rank permutation transformation; support vector machine.

## **I. INTRODUCTION**

One way to perform monitoring and diagnostics for industrial equipment is building a diagnostic engine that reasons over sensor data and outputs the state of the system. The typical steps used here are to read in the sensor data, perform sensor validation, extract features from the sensors, then run selected features through a classifier, perhaps post-process the resulting output by filtering it, and present the result to a decision maker. Applications here range from medical equipment, locomotives to industrial processes, etc.

Another way to recognize faults is through the use of tests that are built into the system. If the conditions of the test are met, output is then typically written out and stored as a text string. Based on the criticality (among other things) of these error logs, maintainers perform remedial action that results in a reset of the condition. In some systems such as aircraft, these error logs are specifically designed to inform operators and maintainers about fault conditions and explicit procedures exist that guide the resulting maintenance.

In other systems, error logs are the byproduct of system developers debugging, not as diagnostic tools for service technicians [1]. Such error logs are not well formatted, and may be polluted with normal status messages. Equipment fault may then generate complex cascades of error messages.

Both approaches have advantages and disadvantages. A dedicated error log message can inform about the exact root cause of a problem. It can be designed to inform about specific system-critical faults that allow the user to react in a timely manner. On the other hand, there is a certain overhead in sensing equipment that goes along with such a dedicated approach. In addition, cascading error logs are not always easy to interpret and it is not always easy to calibrate the sensitivity of these devices correctly. For example, it is not uncommon for machinery to produce thousands of error messages within an hour – while the system is actually considered to behave normally.

In contrast, continuous parametric information from sensors allows one, in principle, to reason over a wider range of faults with a reduced set of sensors. Parametric information also makes it in principle easier to trend information. Such trends may be used as predictive indicators of impending faults. Parametric information is also needed to generate prognostic information, i.e., the estimation of remaining life. However, the tradeoff is in extracting the root cause information from more ambiguous measurements that may be tainted by noise and crosstalk from components that are not of primary interest.

It may be, therefore, advantageous to combine the information from parametric sensors and non-parametric sources (like error logs) to improved fault diagnostics. This paper investigates how one might go about combining parametric with non-parametric information. Specifically, we try to illuminate how non-parametric information can be “parameterized”. We then investigate a particular method for change detection: the rank permutation approach.

This paper is organized as follows. Section II gives an introduction about the rank permutation transformation as the trend shift detection approach and the feature extraction from error log messages. Section III describes the three diagnostic models we employed and compares their results on the real engine data. More discussion on the results is presented in Section IV. Section V concludes the paper.

## **II. DETECTION**

Continuous parametric information from sensors allows one in principle to reason over the system health status. For example, exhaust gas temperature (EGT), fuel flow (WF), engine oil pressure (OIL) and engine core speed (N2) are some of the key parameters measured from sensors. These raw measurements can be directly fed into diagnostic models for evaluating the normal/abnormal condition of the engine. Unfortunately, the raw data space (i.e., using the measurements without further processing) is seldom the best place to attempt to make diagnostic decision. Usually, some feature extraction methods are required in the raw feature space in order to generate more features, which can help capture the characteristic patterns (trends) in the data. In this paper, we employ a

novel data transformation that can be used on its own as a trigger for further investigation in remote monitoring, or in conjunction with various classification techniques for on-wing diagnosis.

### ***Rank Permutation Transformation***

The general approach for fault detection in a time series used here is to compare a test statistic for the “current” data (a few recent points) to the “past” data (some typically larger number of points prior to “current”), possibly with a buffer of unused points in between to make differences more pronounced (Fig. 1).

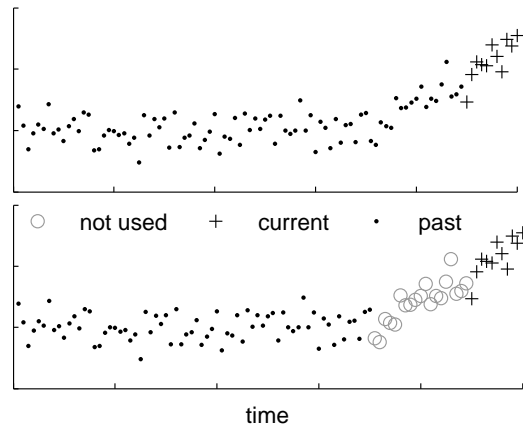


Fig. 1. The general approach to time series fault detection: compare “current” points to “past”. On the top set of axes, all data is used; on the bottom, a small buffer is used to make differences more pronounced.

Using ranks rather than absolute values solves a number of problems. First, the problematic effect of outliers is vastly diminished [2, 3, 4, 5]. The rank transformation is employed to great advantage in nonparametric statistics [4, 5]. The rank distribution for a given number of data can be calculated in advance, making implementation at run time very fast (an important consideration for on-wing applications, which have little computational power to spare).

The idea of using random permutations of the data to develop an exact or near exact probability of occurrence was proposed by Fisher [6]. The principal is illustrated in Fig. 2. The subplot (a) shows the original data. Are the last five asterisk points drawn from the same distribution as the previous dot points? The null hypothesis is that they are not. If the null hypothesis is true, then a statistic (say, the mean) calculated for the asterisk points should be about the same as the same statistic for any five points randomly selected from all of the data. If we keep the data the same, and randomly permute the labels (asterisk or dot), and calculate the “mean of five asterisk points” statistic many times, we get the distribution shown in the subplot (c) (see Fig. 2). This procedure suggests that any five points randomly selected from all points will have a mean as great as the original five only 7.2% of the time. Similarly, any sample of data can be compared to another sample (e.g., using the now vs. recent past approach outlined above), and an exact ((to an arbitrary number of significant figures) probability that the samples differ on any test statistic can be calculated.

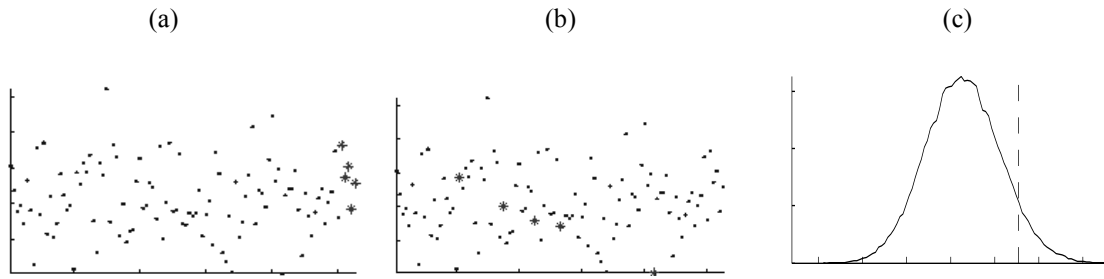


Fig. 2. An illustration of the permutation test. (a) Original data is on the left set of axes. (b) One realization of a random relabeling. (c) The distribution of the test statistic for 100,000 permutations (solid line), and for the original ordering (dashed line).

Putting both ideas (using rank rather than raw data and permutation distributions), the “rank permutation transformation” (RPT) can be used to transform raw, poorly behaved time series features into features that closely represent exact probabilities of occurrence (with the assumption that the error across features is uncorrelated). To calculate RPT, one must first define a small number of points to be the “current” set and a (typically larger) number of points to be the “past” set. The two sets of data are first concatenated and the ranks of the combined data are calculated. The sum of the ranks for the original ordering (the test statistic) for the “current” data is calculated. Next, the data labels (current/past) are randomly permuted, and the test statistic is calculated; this step is repeated many times (e.g., 5,000). The value of the test statistic for the original ordering of the data is compared to the values generated via the permutations, and the probability of the original ordering occurring by chance is calculated. The value of the  $\log_{10}Probability$  (the output of the RPT) is taken to emphasize rare events. For example, the value of 10 ( $\log_{10}10^{10}$ ) means that the rareness of the event is one in 10 billion. The algorithm for calculating RPT can be found in [7].

### ***Feature from Non-Parametric Information***

Another type of data is non-parametric information. In this application, non-parametric information refers to the error log messages generated from the customized on-line built-in tests in the engine system. If the conditions of the test are met, output is then typically written out and stored as a text string (we also call it as message). Based on the different engine components the error log messages inform about, they can be categorized into different types, such as exhaust related, ignition related, oil related and etc. It is not uncommon to have hundreds of error log messages generated from a single flight. There are many ways to “parameterize” the error log messages in order to get some characteristic features, which can be used in diagnostic model, along with parametric data, to assist the engine condition monitoring. Since the parametric data is acquired by taking a snapshot of the sensor measurements during each flight, one way to represent the error log messages is to organize them based on the individual flight. We can count the number of the error log messages occurred during each flight and look for their patterns. It would be more advantageous to count the error log messages based on their categories. This way, the changes of error log messages corresponding to the different engine components/functionalities would be more pronounced.

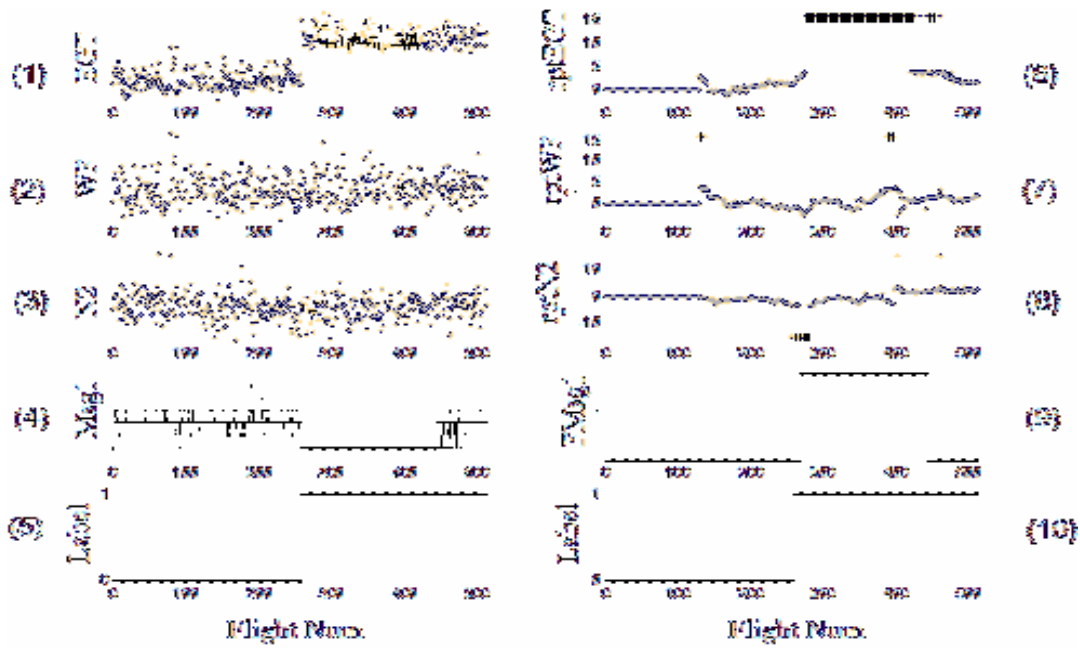


Fig. 3. A case study: data/features from a real engine. Subplots (1)-(3) depict EGT, WF and N2, with black '+' for most likely abnormal points. Subplots (6)-(8) display the outputs from RPT on subplots (1)-(3). Subplot (4) shows the number of *Msg1* in each flight and subplot (9) is the feature extracted from subplot (4). Subplot (5) and (10) are the same, which indicate when the condition of engine is considered as abnormal ('0' as normal and '1' as abnormal).

In Fig. 3, we present the data/features from a real engine. Subplots (1)-(3) display three key gas path parameters, EGT, WF and N2. The black '+' indicate the most probable abnormal points corresponding to their relative big RPT values ( $|rptEGT| > 10$ ,  $|rptWF| > 10$ ,  $|rptN2| > 10$ ) in subplots (6)-(8). The sign of RPT value represents the direction of changes in the parametric data, with 'plus' for up trend and 'minus' for down trend. Subplot (4) counts the number of some error log messages – *Msg1* for each flight of the engine. Subplot (5) and (10) are the same, which indicate when the condition of engine is considered as abnormal based on domain experts' input. Clearly, when we look at the subplot (4), the *Msg1* is quite active until about the 260<sup>th</sup> flight. Then the *Msg1* has not been triggered for the next about 200 flights. Strangely enough, the time when the *Msg1* stops triggering is about the same time when domain experts diagnose the condition of the engine as abnormal. By observing the data of another 31 engines, this type of behavior from error log messages appears to be quite common. In order to capture this behavior of error log messages, we extract a feature from every different category of them although we use the same heuristic to generate them. For example, subplot (9) shows the feature *FMsg1* (the count of error log messages – *Msg1* in a series of flights) generated from subplot (4). The feature shifts its state from '0' to '1' when the error log message suddenly gets turned off after a consistent presence in a certain number of past flights (we can preset the threshold for the number of flights the message has to be active for). When the error message becomes active again for a certain number of flights, the feature switches its state from '1' back to '0'. By using this feature from error log messages, we hope to be able to provide the diagnostic model better information in order to make the more accurate assessment of the engine condition.

The primary goal of diagnostics is to identify the abnormal condition of the system as early, accurate and consistent as possible. If there is anything unusual going on and it is exhibited from either parametric or non-parametric information, the diagnostic model is expected to recognize it. Once the model identifies the abnormal behavior of the engine, and if no follow-up remedial/maintenance action is done, the abnormal signatures of the engine usually will not change. Therefore, the condition assessment from the model should remain unchanged. In this paper, we build three diagnostic models on the features extracted from the parametric and non-parametric data. In a forwarding mode, given the data accessible at each flight, the diagnostic model makes an assessment (normal or abnormal) about the condition of the engine. At the end, we compare the decisions from the three models against each other and with the assessment by domain experts.

### III. RESULTS

A suite of diagnostic models is built up to integrate selected features to make a diagnostic decision. These models are neural network (NN), random forest (RF) and support vector machine (SVM). Each model has its advantages and disadvantages. Since they are quite different in principle, they give different decisions about the condition of the engine. By providing the diversities in the diagnostic models, it creates the opportunities of fusing the decisions from each model together to make a better assessment (although the decision fusion from the different models is beyond the scope of this paper). The overall fused decision will be better than the output from the individual model alone.

#### *Neural Networks*

Neural networks, usually composed of simple elements (called neurons) operating in parallel, are well known for their function approximation and pattern recognition performance [8, 9, 10]. They have been successfully applied across an extraordinary range of domains, in area of finance, engineering, medicine, physics and etc. In this application of aircraft engine fault diagnostics, we employ neural networks as a classifier for the condition of the engine. Given the complexity of the problem, a simple linear network model is chosen. A linear network has one layer of  $S$  neurons connected to  $R$  inputs through a matrix of weight  $\mathbf{W}$ . The network output is  $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$ , where the input  $\mathbf{x}$  is  $R$  by 1 vector, weight  $\mathbf{W}$  is  $S$  by  $R$  matrix, bias  $\mathbf{b}$  is  $R$  by 1 vector and  $\mathbf{y}$  is  $S$  by 1 vector. Since the diagnostic model only needs to give a classification of engine condition as output,  $S$  equals to 1 here. Given a labeled dataset,

$$D = \{(x_i, y_i)\}_{i=1}^l, x_i \in X \subset R^d, y_i \in Y = \{-1, +1\} \quad (1)$$

The linear network can be trained to minimize the mean square error (MSE)

$$MSE = \frac{1}{l} \sum_{i=1}^l e_i^2 = \frac{1}{l} \sum_{i=1}^l (y_i - \hat{y}_i)^2, \hat{y}_i: \text{output of the network}, \quad (2)$$

by adjusting the weights and biases. Unlike the other types of the neural networks, which usually require training, the linear networks can also be designed directly if input/target vector pairs are known. The weights and biases can be calculated to minimize the  $MSE$  on the training dataset. This has been implemented and become a function of MATLAB Neural Network Toolbox.

### **Support Vector Machine**

We also consider here a Support Vector Machine (SVM) as the classification engine [11, 12]. A SVM is a maximum margin linear classifier in a high dimensional feature space  $\Phi(x)$  defined by a positive definite kernel function  $k(x, x')$  specifying an inner product in the feature space  $\Phi(x)$ .  $\Phi(x') = k(x, x')$  given a labeled data set as in Eq. (1). The Gaussian radial basis function

$$k(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \quad (3)$$

can be used as a kernel function. The function implemented by a support vector machine is given by

$$f(x) = \left\{ \sum_{i=1}^l \alpha_i y_i k(x_i, x) \right\} - b, \quad (4)$$

where the optimal coefficients  $\alpha$  can be maximized by the functional

$$W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j k(x_i, x_j) \quad (5)$$

in the non-negative quadrant  $0 \leq \alpha_i \leq C, i = 1, \dots, l$  subject to the constraint

$$\sum_{i=1}^l \alpha_i y_i = 0. \quad (6)$$

$C$  controls the compromise between maximizing margin and training error. The optimization conditions (Karush-Kuhn-Tucker conditions) are:

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i f(x_i) \geq 1, \\ 0 < \alpha_i < C &\Rightarrow y_i f(x_i) = 1, \\ \alpha_i = C &\Rightarrow y_i f(x_i) \leq 1. \end{aligned} \quad (7)$$

A set of Lagrange multipliers  $\alpha^0$  satisfies these conditions. The bias  $b$  is set s.t. the second optimization condition is met. Input patterns for feasible Lagrange multipliers are called support vectors [13].

### **Random Forest**

Random Forest [14] is another classifier we use here. It is a classification method that applies bagging [15] to a variation of classification trees [16]. A standard classification tree is constructed by splitting the data on the best feature of all possible features at each node. For RF, only a randomly selected subset (chosen always from the full set) of features is eligible to split each node. Moreover, each individual tree is constructed on a bootstrap sample of the data. Finally, in contrast to standard classification trees, the individual RF trees are not pruned; rather they (typically) are grown to 100% node purity. Although typically hundreds of trees are developed, RF's are very quick to train (e.g., much faster than neural networks for a given data set and processor). Predictions are made by aggregating the predictions of the ensemble (majority vote for classification or averaging for regression). Random forest generally exhibits a substantial performance improvement over the single tree classifier such as CART and C4.5. It yields generalization error rate that compares favorably to Adaboost, yet is more robust to noise. The RF implementation used here was by Breiman et al. [17].

One of the beneficial features of the RF algorithm is its use in variable selection. To estimate the importance of a particular variable, the values of that variable are permuted, and predictions of the out-of-bag cases for each tree are made using the permuted data. For a particular case, the margin is the proportion of votes for its true class minus the maximum of the proportion of votes for each of the other classes. The RF importance measure for a particular variable is the average lowering of the margin across all cases when that variable is randomly permuted. This value can be calculated quite quickly for each variable in turn.

### Experiment

For the experiment, we used data from 32 time series of engine data and error logs. At the beginning, the full set of 26 data features (from parametric and non-parametric data) were examined using RF. By using RF variable importance measure (averaged over the 32 time series), the original feature space was reduced from 26 to the most important 9 features, in which include the raw measurements from 3 sensors, the RPT value from the 3 sensor measurements and the features from 3 types of error log messages. Then, all the three models used the same set of 9 features as input for classification. A leave-one-out (LOO) approach was employed, where we used 31 time series for training and one time series for testing. This one test time series and the training set were then rotated such that 32 different classifiers were built. The performance was averaged across the different test runs.

Model	Confusion Matrix				Overall Accuracy
	True Positive	False Negative	False Positive	True Negative	
NN	64%	36%	7%	93%	87%
SVM	60%	40%	13%	87%	81%
RF	90%	44%	13%	87%	80%

Table I. The diagnostic results on real engine data from NNs, SVM and RF model.

Table I compares the performance of the three models in terms of their best overall accuracy on the 32 time series of engine data using the LOO procedure. Each model has different performance on the components of the confusion matrix. Depending on the primary goal of the diagnostic system, one can weight the performance of the model differently based on the confusion matrix and choose the most appropriate model. The ROC curves of the three models are shown in Fig. 4, with the asterisk points corresponding to the best overall accuracy in each model.

## IV. DISCUSSION

When we evaluated the performance of each model, we counted each individual incorrect classification as either a false positive or false negative. This may lead to overly harsh assessment because an early prediction – while technically desired – is counted as a false positive. It would be more reasonable to use a different metric to assess the diagnostic decisions from the models, which does not take the early false positives close to the start

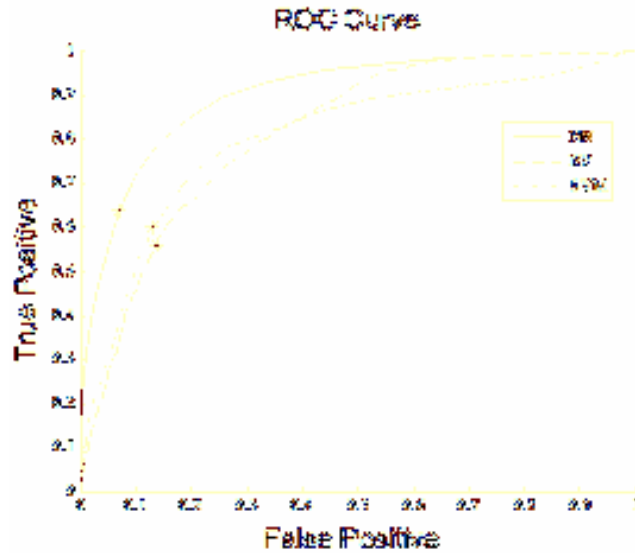


Fig. 4. ROC curves of the three (NN, RF and SVM) diagnostic models.

of the engine abnormal condition as misclassification. The new evaluation metric should also give relative small penalties for decisions which do not recognize the abnormal condition immediately as long as the model catches the fault soon after. Fig. 5 illustrates how a heuristic evaluation metric can reasonably improve the performance of the diagnostic models in detecting the engine abnormal conditions. In Fig. 5, the diagnostic decisions from the three models on one time series of engine data are plotted out against the condition assessment by domain experts. The solid line represents the starting time when the engine is considered to have the abnormal condition, as indicated in the 'target' subplot. In the heuristic evaluation metric, the fault positive decisions from the SVM model (see Fig. 5) before the solid line can be taken as the beneficial early fault detection.

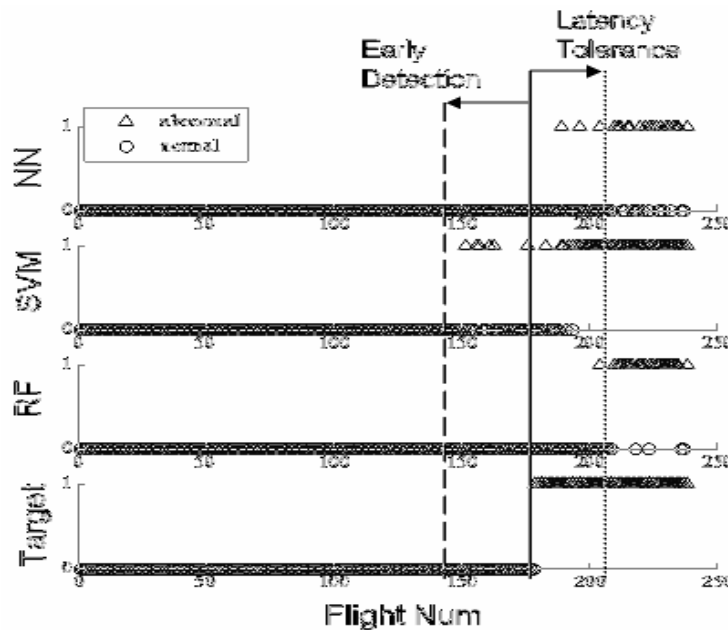


Fig. 5. An illustration of a heuristic performance evaluation metric.

Also, the latency detection from the RF model (see Fig. 5) should be less penalized given a reasonable latency diagnosis boundary, as indicated by the dotted line. Using such a heuristic evaluation metric, the engine condition assessments from the three diagnostic models can be reasonably improved in terms of detecting the engine abnormal condition event. As part of the future work, a data fusion scheme can be explored to integrate the decisions from these three reasoning models to enhance the overall diagnostic performance.

## V. SUMMARY

A suite of classifiers was built on both parametric and non-parametric information to diagnose the abnormal conditions of aircraft engine. We employed a novel trend shift detection approach on the parametric sensor measurement to detect when the engine performance data start to change away from their nominal level. Features were also developed from the error log messages to capture the characteristic patterns of the messages. Promising results were achieved from the three diagnostic models (NNs, SVM and RF). The performance of the overall model will be further explored by incorporating advanced performance evaluation metric and a decision fusion paradigm.

## REFERENCES

- [1] G.C. Cawley, Support Vector Machine Toolbox v0.50 beta, <http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox>, University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ, 2000.
- [2] P. Good, *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*, Springer-Verlag, New York, 1993.
- [3] H. Mann, and D. Whitney, "On a test of whether one of two random variables is stochastically larger than the other", *The Annals of Mathematical Statistics*, 18(1), pp. 50-60, 1947.
- [4] F. Mosteller, and J. Tukey, *Data Analysis and Regression: A Second Course in Statistics*. Addison Wesley, Reading, MA, 1977.
- [5] F. Wilcoxon, "Individual comparisons by ranking methods", *Biometrics*, 1, pp. 80-83, 1945.
- [6] R. Fisher, "The logic of inductive inference." *Journal of the Royal Statistical Society*, 98, pp.39-54, 1935.
- [7] N. Eklund, and K. Goebel, "Using Neural Networks and the Rank Permutation Transformation to Detect Abnormal Conditions In Aircraft Engines", *Soft Computing in Industrial Applications, SMCia/2005, Proceedings of the 2005 IEEE Mid-Summer Workshop on*, pp. 1-5, 2005.
- [8] S. Haykin, *Neural Networks: a comprehensive foundation*, 2<sup>nd</sup> Ed., Prentice Hall, New Jersey, 1999.
- [9] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University, Press. 1995.
- [10] X. Hu, G. Clark, M. Travis, J. Vian, and D. Wunsch, "Aircraft Cabin Noise Minimization via Neural Network Inverse Model", *Neural Networks, 2005, Proceedings of the International Joint Conference on*, vol.4, pp.3001-3006, 2005.
- [11] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
- [12] B. Cheetham, P. Cuddihy, and K. Goebel, "Applications of soft CBR at GE", *Soft Computing in Case Based Reasoning*; Eds: S. Pal, T. Dillon, and D. Yeung; Springer Verlag, London, 2001; (ISBN 1-85233-262-X), pp. 335-365, 2001.
- [13] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization", *Advances in Kernel Methods - Support Vector Learning*, (Eds) B. Scholkopf, C. Burges, and A. J. Smola, MIT Press, Cambridge, Massachusetts, chapter 12, pp. 185-208, 1999.
- [14] L. Breiman, "Random forest", *Machine Learning*, 45(1), pp. 5-32, 2001.
- [15] L. Breiman, "Bagging predictors", *Machine Learning*, 24(2), pp. 123-140, 1996.
- [16] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1984.
- [17] L. Breiman, and A. Cutler, Random Forest Toolbox Version 5.1, [http://www.stat.berkeley.edu/users/breiman/RandomForests/cc\\_software.htm](http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_software.htm), University of California at Berkeley.