Feature Extraction and Ensemble Decision Tree Classifier in Plant Failure Detection

Cong Xie, Donglin Yang, Yixiang Huang, and Donglai Sun

Maxtropy Technology, Shanghai, 201199, China
xiecong,yangdonglin,huangyixiang,sundonglai@maxtropy.com

Abstract

This paper describes a set of algorithms used to tackle the plant prognostic problem provided in the IEEE 2015 PHM Data Challenge. The task is to detect failure events by analyzing a dataset including sensor measurements and control reference signals of multiple plants without prior knowledge. There are two main difficulties lies in the data challenge. One is to identify which of the faults will occur. And the other is when the fault is going to happen. In this study, the authors tried to transform the task issue into a classification problem by three key steps including: 1) data cleansing and event time alignment; 2) feature extraction; 3) application of the ensemble decision tree classifiers. Results show that the proposed data-driven methods can effectively detect several types of the failure events, which may be promising in the real world plant prognostic applications.

1. Introduction

In recent years, fault detection (Isermann, 1984; Isermann & Ballé, 1997; Samy, Postlethwaite, & Gu, 2011; Isermann & Isermann, 2011; Gao, Cecati, & Ding, 2015) has attracted increasing attention in both academic and industrial fields. The complexity of plants and large amount of integral subsystems necessitates automated prognostic system, which is used to replace the human supervision. Researchers and engineers now confront challenges which require advanced technologies rather than the traditional model-based methods.

The approaches to fault detection can be generally classified into two categories: model-based methods and data-driven methods. The traditional model-based methods (Venkatasubramanian, Rengaswamy, Yin, & Kavuri, 2003; Isermann, 2005; Frank, 1990) are based on the physical models of the systems and the related experience and expertise. However, the complexity of modern industrial systems sets obstacles for devising a practical model. Furthermore, in some cases we simply have no idea about the structure of the system. What we have is the mere data recorded from the sensors, which motivates us to use data-driven methods (Yin, Wang, & Karimi, 2014; Schwabacher, 2005; Qin, 2009). Data-driven methods tend to introduce technologies of machine learning and data science to prognostics.

This paper presents the methods developed by the team Maxtropy for the IEEE 2015 PHM Data Challenge. The prognostics topic focuses on the operation of a plant and the capability to detect plant failure events. The dataset is composed of the following three parts:

(a) time series of sensor measurements and control reference signals for each of a number of control components of the plant (e.g. 6 components);

(b) time series data representing additional measurements of a fixed number of plant zones over the same period of time (e.g. 3 zones), where a zone may cover one or more plant components;

(c) plant fault events, each characterized by a start time, an end time, and a failure code.

Only faults of type 1-5 are of interest, while code 6 represents all other faults not in focus. The frequency of measurements is approximately one sample every 15 minutes, and the time series data spans a period of approximately three to four years. The goal is to predict the beginning time and end time of failure events of types 1-5. More detailed information could be found on the website of PHM Society (PHM Society, 2015). The dataset can be downloaded from NASA Ames Prognostics Data Repository (J. Rosca, 2015).

In this paper, data-driven methods are adopted under the circumstance that we have no prior knowledge about the structure or physical characters of the plants. More specifically, we extract several unordered features from the raw data. Pieces of time series are then sliced. Finally, we train classifiers and predict to which kind of fault each piece belongs. We use ensemble decision tree methods, including RF (Random Forest) and GBDT (Gradient Boosting Decision Tree), as the classifiers.

The rest of the paper is organized as follows. In Section 2, we formally define the problem to be solved. The detailed solution of our team is proposed in Section 3. Empirical results are presented in Section 4. Finally, we conclude the paper in Section 5.

2. PROBLEM FORMULATION

In this section, we formally describe the problem in PHM 2015 data challenge.

2.1. Notations

We use bold letters for vectors, and normal fonts for scalars, sets. Bold capital letters are used for matrices. All the vectors in this paper are column vectors. And S^T denotes the transpose of matrix S. $1(condition)$ denotes the indicator function, whose value is 1 when the condition is satisfied and 0 otherwise. Each observation is represented as a pair of features and a time stamp: $(\mathbf{x}, t)$, where $\mathbf{x} = [x_1, \ldots, x_k]^T$ is a k-dimensional feature vector and t represents the time when such sample is observed. A fault event is composed of a pair of time stamps and a fault label: (t^{start}, t^{end}, y), where t^{start} is the beginning time of the fault, t^{end} is the end time of the fault, and $y \in \{1, 2, 3, 4, 5\}$ is the corresponding fault label. The sequence of observations in dataset X corresponding to a fault event (t^{start}, t^{end}, y) is $\mathbf{s} = \big\{(\mathbf{x}, t)\big|(\mathbf{x}, t) \in X, t \geq t^{start}, t \leq t^{end}\big\}$.

2.2. Problem Definition

Two groups of data are provided in PHM 2015 Data Challenge: component sensor measurements along with control reference and zone sensor measurements. Although both groups of data are supposed to be useful, we only adopt the component sensor measurements along with control reference to simplify our model. Hence, the raw data have totally 9 features: 1 integer indicates the component number, 4 sensor measurements and 4 control reference. We represent the data of the a specific plant as $X^{raw} = \big\{(\mathbf{x}_i, t_i)\big|i \in \{1, \ldots, N\}\big\}$, where N is the number of observations of the plant. The set of corresponding fault events can then be represented as $E = \big\{(t_i^{start}, t_i^{end}, y_i)\big|i \in \{1, \ldots, M\}\big\}$, where M is the number of fault events of the plant. $E^{T-} = \big\{e = (t^{start}, t^{end}, y)\big|e \in E, t^{end} \leq T\big\}$ denotes all the events before T. $E^{T+} = \big\{e = (t^{start}, t^{end}, y)\big|e \in E, t^{end} \geq T\big\}$ denotes all the events after T. To simplify the problem, we assume that $E^{T-} \cap E^{T+} = \emptyset$.

For a specific plant, we have the data X^{raw} and complete E^{T-} along with a specific T. E^{T+} is incomplete. The task is to find all the missing elements in E^{T+}.

3. METHODOLOGY

In this section, we propose the solution to the problem defined in Section 2.2.

3.1. Feature Extraction

The first step is to extract useful features from the raw data to facilitate the detection. Note that the fault events can overlap. Hence, we draw an assumption that the indicator of a fault event may involve a subset of all the components. Furthermore, such indicator may be independent on the order of the components.

The simplest way of feature extraction is to connect the raw data vector in a specific order. In particular, raw data vectors are aligned by the time stamps. Observations with nearly the same time stamps are collected into a group and combined into a longer vector, which is the feature vector. The combination is according to a specific order of component number. The component numbers are then useless and can be removed from the feature vector. In addition, some other features can also be combined into the feature vector introduced above. The time series of fault events follow a periodical pattern. Month, day, weekday and hour can be extracted from time stamps. Then features are extracted by getting the maximum, minimum, average and standard deviation of each column of S from observations in the same month. Similarly for the same day, weekday and hour. Another similar aspect is that plant fault events are independent of one another but a fault is possible to be dependent of data i inside a three hour time window before the fault start time. Then features are extracted by getting the maximum, minimum, average and standard deviation of each column of S from observations in the one hour. Similarly for two hours and three hours time window. (Faloutsos, Ranganathan, & Manolopoulos, 1994; Huang et al., 1998)

According to our assumption, the model should be independent on the order of the components. To fulfil this goal, we can simply consider all the permutations of the observations in the same group. All such permutations are then combined into feature vectors. Such way of feature extraction does make sense. Nevertheless, when the number of components of a plant is large, the number of data after feature extraction grows dramatically. For example, if there are 10 components in some plant, $10! = 3,628,800$ permutations should be considered, which makes the problem intractable.

Another way rather than permutes the components is to draw histograms from each dimension of the group of raw data which are aligned by the time stamps and then connect the histograms into a longer vector. Note that this solution can also remove the order of components from the features. However, the bins of the histograms should be carefully selected via inspecting each dimension of the observations. Small number of bins may eliminate the discrepancy between failures and normal status. Too many bins may result in overfit or bad generalization.

In our solution, a list of all unique values is drawn out of each dimension of raw data. One out of every 8 elements in the list is selected as a bin. A region of higher density of elements also possess more bins.

Now we formally define the feature vector. For a specific sequence of k observations $S = \{(\mathbf{x}_i, t_i) | i \in \{1, \ldots, k\}\}$, we construct a corresponding $k \times 8$ matrix $\mathbf{S} = [\mathbf{x}_1, \ldots, \mathbf{x}_k]^T$, each row of which is an observation. Note that we assume that the component numbers are already removed from the observations. For a certain dimension $j \in \{1, \ldots, 8\}$, we have the set of bins $B_j = \{b_{j,1}, \ldots, b_{j,p_j+1}\}$. The corresponding histogram of the jth dimension is a $p_j \times 1$ vector $\mathbf{f}_j = [f_{j,1}, \ldots, f_{j,p_j}]$, where $f_{j,i} = f'_{j,i} / \sum_{i \in \{1,\ldots,p_j\}} f'_{j,i}$ and $f'_{j,i} = \sum_{l \in \{1,\ldots,k\}} \mathbf{1}(b_{j,i} \leq \mathbf{S}_{l,j} < b_{j,i+1})$ for $\forall i \in \{1, \ldots, p_j\}$. The final feature vector of S is simply the combination of all $\mathbf{f}_j$, which is $[f_{1,1}, \ldots, f_{1,p_1}, \ldots, f_{8,1}, \ldots, f_{8,p_8}]^T$.

3.2. Ensemble Decision Tree Classifier

Ensemble methods are powerful tools for classification. In our solution, we adopt decision tree classifier to predict the type of fault. More specifically, we use RF (Random Forest) and GBDT (Gradient Boosting Decision Tree) as the classifiers.

RF (Random Forest) build an ensemble of classifiers and it takes advantage of two powerful machine-learning techniques: bagging and random feature selection (Liaw & Wiener, 2002; Breiman, 2001). Bagging constructs new training sets by resampling from the original data set by randomly select k samples. Note that the sample selected will not be removed from the data set in the next draw. Bootstrap sampling technique makes some of the training samples be chosen more than once while some others will not be chosen at all in a new training set. For each training set, instead of using all features, RF randomly selects a random subset of the input features to split at each splitting node when growing a tree. Since only a subset of the features is utilized at each node, computational load of RF is comparatively light. To assess the prediction performance, an out-of-bag (OOB) method can be used. For each training set, one-third of the samples are randomly left out and two-thirds of the samples are used for building a tree. For accuracy estimation, votes for each sample in OOB samples can be used to estimate the performance of prediction. In the end, a simple majority vote is taken for prediction. (Svetnik et al., 2003)

GBDT (Gradient Boosting Decision Tree) is a data mining technique that has achieved considerable success in data mining (Dietterich, 2000; J. Friedman, Hastie, Tibshirani, et al., 2000; J. H. Friedman, 2001, 2002). There are examples of gradient boosting applications in other fields including refinement of classification tree analysis in a remote sensing problem (Lawrence, Bunn, Powell, & Zambon, 2004), discrimination of freshwater residency in a coastal fishery from scales collected from subadult fish (McCulloch, Cappo, Aumend, & Müller, 2005), microscopy image analysis of bread (Lindgren & Rousu, 2002), graphical estimation of a slate deposit (Diener et al., 2004), and calibrating spectroscope measurements of organic chemicals in plant samples (Shepherd, Palm, Gachengo, & Vanlauwe, 2003).

Boosting creates a series of decision trees which together form a single predictive model. Trees are built sequentially from pseudo-residuals, which is the gradient of the loss function of the previous tree. At each iteration, a tree is built from a random sampling of the original data set, producing an incremental improvement in the model. This process is similar to a bootstrap technique in that many trees are generated. With each successive tree, it is hoped that gradient boosting will reduce the error. Using only a fraction of the training data increases both the computation speed and the prediction accuracy, while also helping to avoid over-fitting the data. An advantage of stochastic gradient boosting is that it is not necessary to select predictor variables ahead of time or transform predictor variables. It is also resistant to outliers, as the steepest gradient algorithm stresses points that are close to the correct classification (Moisen et al., 2006; B. Roe et al., 2005).

Algorithm 1 Training of a plant

```
Input: $X^{raw}$, $E^{T-}$, set of bins $B = \{B_1, \ldots, B_8\}$
Output: A function $label := predictor(sample)$
 1: $S = \emptyset$
 2: for all $e \in E^{T-}$ do
 3:     Extract the feature vector $\mathbf{f}$ from $e$ by using $b$
 4:     Add $\mathbf{f}$ into $S$
 5: end for
 6: for all $e \notin E^{T-}$ and before the last event in $E^{T-}$ do
 7:     Extract the feature vector $\mathbf{f}$ from $e$ by using $B$
 8:     Add $\mathbf{f}$ into $S$
 9: end for
10: $predictor = train\_trees(S)$
```

3.3. Algorithm

Now we can formally describe our algorithms. The solution is mainly separated into two parts: training and prediction. For training, features are extracted from all the fault events in X^{raw} indicated by E^{T-} as well as an adequate number of normal (without any fault) events. All these samples are used to train the classifier. The detailed algorithm is shown in Algorithm 1. Note that $train\_trees$ can be either RF or GDBT. For prediction, moving windows of different lengths are used to draw events from X^{raw} after the last event in

E^{T-}. All such events are labelled by the classifier trained before. Overlapping fault events with same label will then be combined into a single event. Finally we recover the set E^{T+}. Detailed algorithm is shown in Algorithm 2.

Algorithm 2 Prediction of a plant

```
Input: $X^{raw}$, $E^{T-}$, set of bins $B = \{B_1, \ldots, B_8\}$,
    $predictor$, a set of lengths of events $l$
Output: $E^{T+}$
 1: $S = \emptyset$
 2: An empty label 0 is initially assigned to each event $e$
 3: for all $e = (t^{start}, t^{end}, 0)$ after the last event in $E^{T-}$ of
    length in $l$ do
 4:     Extract the feature vector $\mathbf{f}$ from $e$ by using $b$
 5:     $label = predictor(\mathbf{f})$
 6:     $e^{new} = (t^{start}, t^{end}, label)$
 7:     Add $e^{new}$ into $S$
 8: end for
 9: repeat
10:     for all $g \in S$ do
11:         for all $h \in S$ and $h \neq g$ do
12:             if $g$ and $h$ overlap and have same label then
13:                 Combine $g$ and $h$ into a single event $e_{gh}$
14:                 Delete $g$ and $h$ from $S$
15:                 Add $e_g h$ into $S$
16:             end if
17:         end for
18:     end for
19: until No overlapping events with same label are found
20: $E^{T+} := S$
```

Note that each plant is processed separately according to the way we normalize the histograms, though samples from different plants have the same number of features. Some other kinds of features may prevent us from differentiating samples from various plants, but we will not discuss such details in this paper.

4. EXPERIMENT

In this section, we present the empirical results of our methods. We compare the performance of RF and GDBT. Furthermore, we report the predicting result of each plant according to the scoring criterion.

4.1. Dataset

The dataset we use is presented in Table 1. Totally 32 plant is used. Plant 2 is not included for the missing of many fault events after a certain time point. Fault events of the remaining plants are complete. We separate the data of each plant into two parts: training data and testing data. In Table 1, # samples 1 and # faults 1 denotes the numbers of observations and faults before a specific time point, which are used for training. And # samples 2 and # faults 2 denotes the numbers of observations and faults after such time point, which are used for testing.

4.2. Classification

In this section, we evaluate the classifiers. We simply extract all the fault events and randomly pick up 2000 normal events out of each plant for testing. The number of ensemble trees is 256 for each classifier. The ensemble method we use for GDBT is called *RUSBoost* in MATLAB. Accuracy, recall and true negative of both algorithms are shown in Figure 1, 2 and 3. Accuracy is the percentage of events which are correctly labelled. Recall is the percentage of fault events which are correctly labelled. True negative is the percentage of normal events which are correctly labelled. RF shows competitive performance in recall. And RF avoids more false positive. Generally, RF overshadows GDBT. Note that for RF, the number of normal events for training must be carefully tuned, while such number affects the performance of RUSBoost slightly.

Table 1. Dataset

Plant #	# samples 1	# faults 1	# samples 2	# faults 2
1	1728	1155	1546	1157
3	1339	441	3148	442
4	1724	914	1811	916
5	574	389	1269	391
6	1619	1152	2750	1153
7	3355	2124	3297	2126
8	1614	1064	1660	1065
9	917	734	1262	735
10	748	486	748	487
11	221	23	78	25
12	862	466	522	468
13	2047	1824	2245	1825
14	1425	936	1671	937
15	2395	1472	2306	1473
17	498	238	2311	239
18	1647	1399	1951	1401
21	1462	182	788	183
23	1488	652	2393	653
24	380	319	1165	320
25	1820	1275	1525	1277
27	1401	949	2385	950
28	382	207	535	208
29	1180	559	856	561
30	1977	1136	1248	1138
31	375	141	187	142
32	1815	1565	1833	1566
33	548	415	535	416
34	951	697	1578	698
35	1483	1268	1466	1269
36	925	619	754	620
39	1561	689	1301	690
40	1679	1592	1841	1594

4.3. Scoring

The scoring criterion is as follows:

$$Score = TP \times 10 - MC \times 0.01 - FP \times 0.1 - FN \times 0.1,$$

where TP is the number of faults identified correctly with the start and end time estimated within ± 1 hour, MC is the

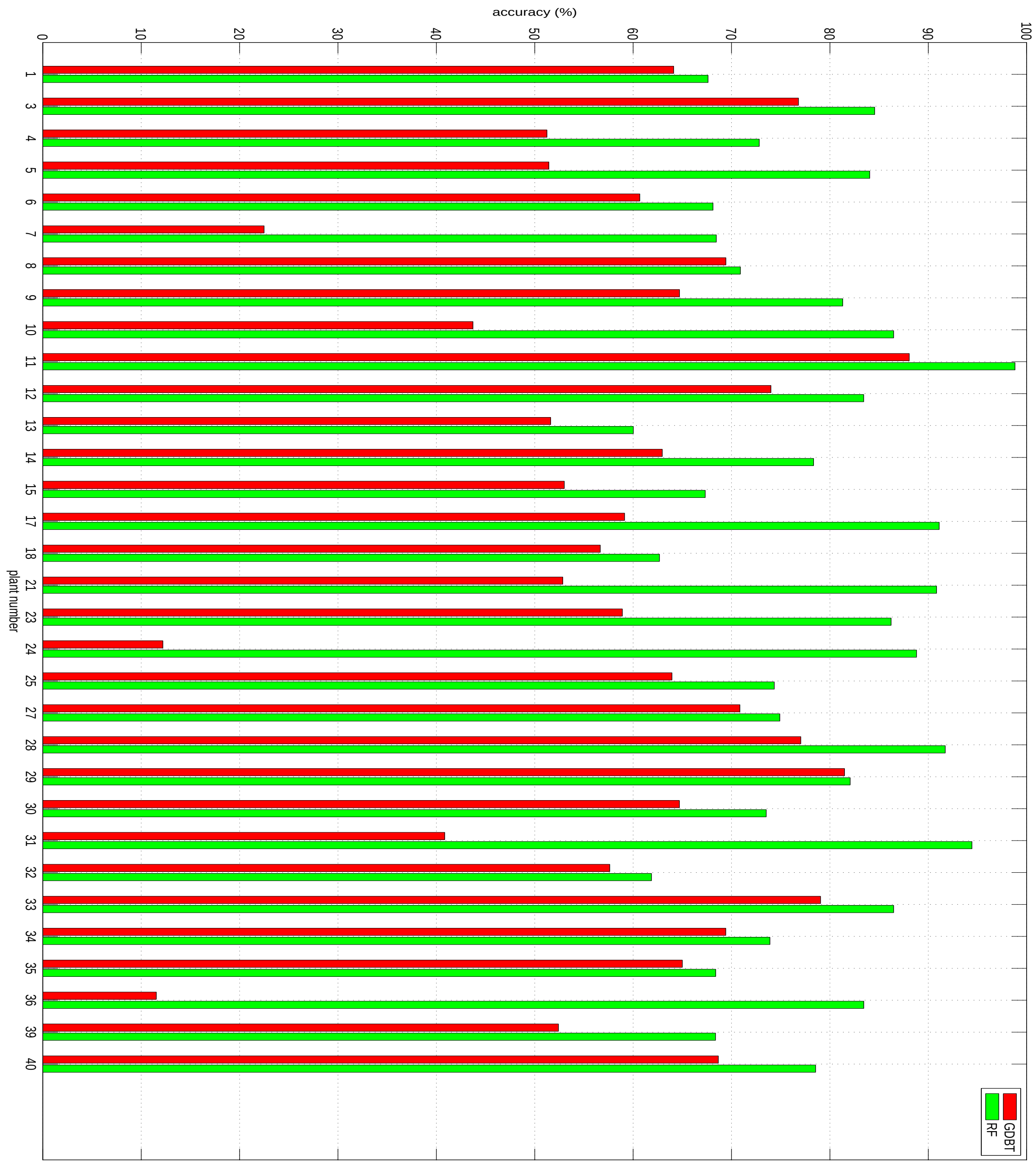

Figure 1. Classification result: accuracy

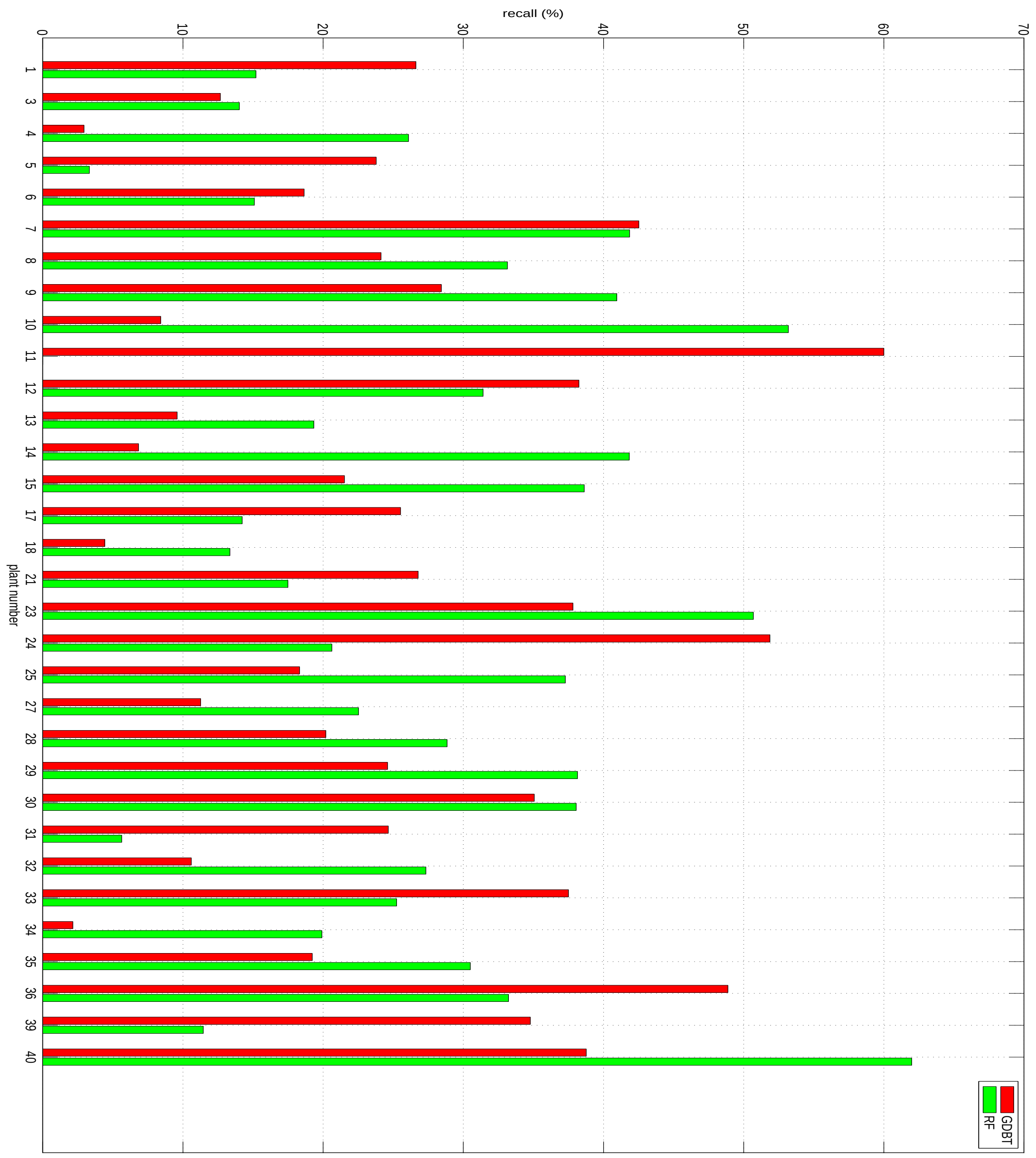

Figure 2. Classification result: recall

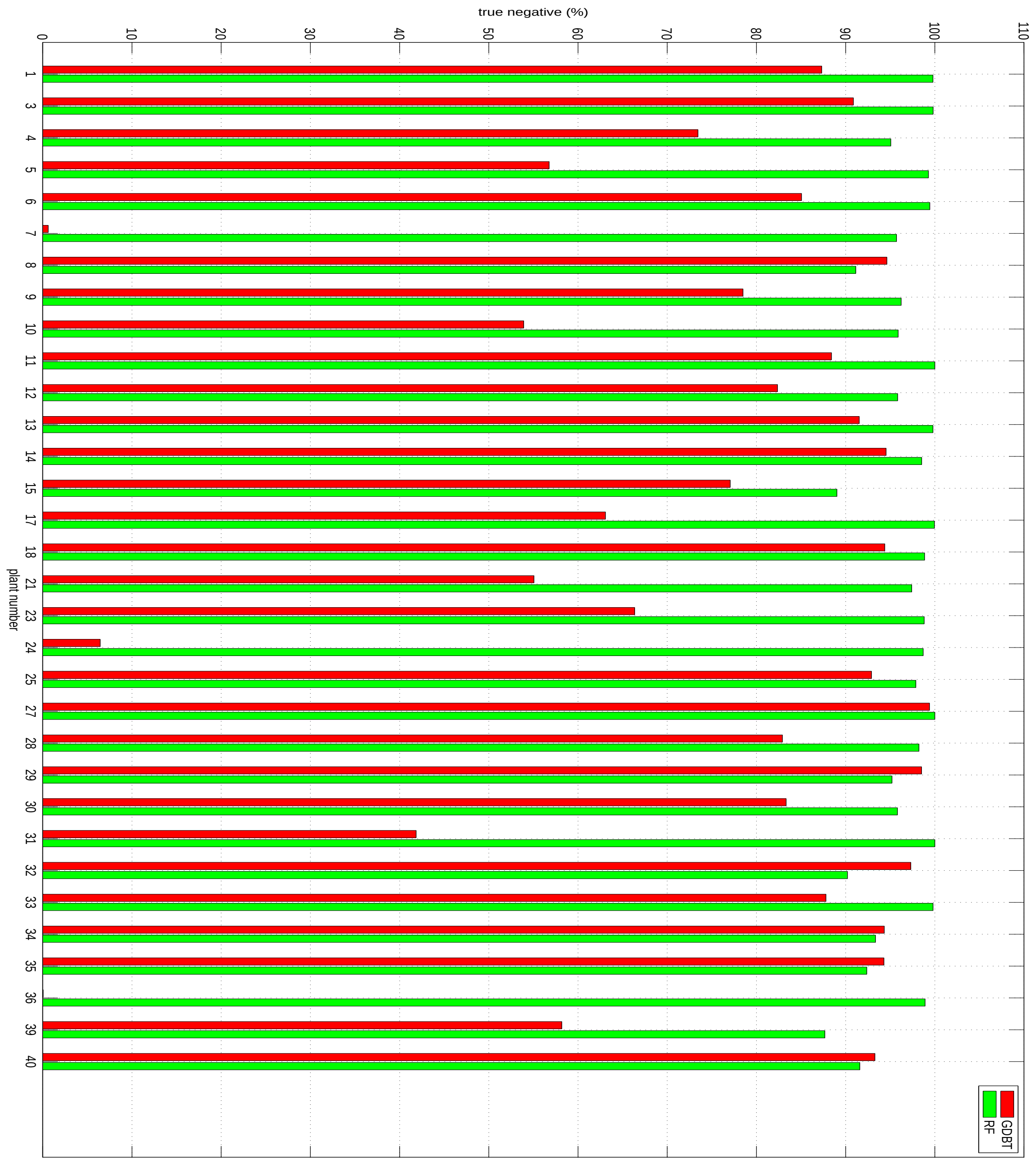

Figure 3. Classification result: true negative

number of faults identified with correct start and end times but with the wrong fault code, FP is the number of faults identified that did not actually occur in the data, FN is the number of faults in the real data that were not identified.

Table 2. Scoring GDBT

Plant #	TP	MC	FP	FN	Score
1	307	1302	14163	616	1579.08
3	206	571	30028	222	-970.71
4	17	578	10226	739	-932.28
5	158	527	36575	146	-2097.37
6	277	1118	3842	545	2320.12
7	272	3151	41052	1016	-1518.31
8	34	257	1315	936	112.33
9	439	1377	17014	201	2654.73
10	137	739	13654	289	-31.69
11	3	14	1569	18	-128.84
12	93	517	4368	256	462.43
13	73	1947	7398	1169	-146.17
14	31	481	2594	740	-28.21
15	325	855	12262	1016	1913.65
17	42	241	12036	145	-800.51
18	42	417	3109	1210	-16.07
21	26	270	8762	87	-627.6
23	201	617	13805	318	591.53
24	93	494	38880	169	-2979.84
25	52	1078	7417	851	-317.58
27	82	501	1853	675	562.19
28	11	224	7467	134	-652.34
29	127	586	6174	337	613.04
30	212	1574	5987	568	1448.76
31	43	252	1717	60	249.78
32	104	1474	6471	941	284.06
33	111	519	8842	236	197.01
34	21	140	1421	648	1.7
35	84	591	2937	992	441.19
36	204	1267	40601	133	-2046.07
39	3	295	5703	586	-601.85
40	70	502	3365	1376	220.88

The result of GDBT is shown in Table 2. And the result of RF is shown in Table 2. It can be seen that RF outperforms GDBT in most cases. Although changing some parameters of GDBT may help to improve the performance, RF works well without much effort of tuning.

The major problem of our solution is that it is still difficult to tell apart the overlapping fault events. And in many cases, a long-time fault event may be detected as many short events, which results in many false positives. A more effective linking strategy is needed. Furthermore, the bins used for feature extraction may not be the optimal one. Further tuning may be necessary.

5. CONCLUSION

In this paper, we have proposed a solution to the problem of the IEEE 2015 PHM Data Challenge. A useful feature extraction technology as well as ensemble decision tree classifiers have been utilized. The empirical results have also been presented. More efforts will be taken in diminishing the false positives in our future work.

Table 3. Scoring RF

Plant #	TP	MC	FP	FN	Score
1	494	2406	10307	272	3858.04
3	101	189	4864	320	489.71
4	548	1233	11636	276	4276.47
5	102	200	9056	251	87.3
6	431	1051	4485	502	3800.79
7	678	2405	14246	1046	5226.75
8	294	1465	11929	487	1683.75
9	411	920	6208	258	3454.2
10	126	396	3450	350	876.04
11	0	0	0	25	-2.5
12	106	539	4383	247	591.61
13	442	1925	6940	925	3614.25
14	269	979	6270	540	1999.21
15	503	1932	14847	518	3474.18
17	19	56	963	210	72.14
18	511	2035	8752	490	4165.45
21	45	69	911	129	345.31
23	303	892	10373	258	1957.98
24	46	120	2444	261	188.3
25	430	1357	11505	615	3074.43
27	177	657	1980	639	1501.53
28	39	126	1685	156	204.64
29	186	666	6721	284	1152.84
30	344	1327	5856	597	2781.43
31	4	11	49	136	21.39
32	407	1887	14884	781	2484.63
33	33	51	726	371	219.79
34	255	648	7342	330	1776.32
35	336	1666	11715	657	2106.14
36	343	723	5284	226	2871.77
39	89	532	8290	513	4.38
40	634	1535	13093	794	4935.95

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers and editors for their helpful comments, as well as the organizers of the PHM data challenge competition for their efforts to give the opportunity for us to present the potential solutions to industrial applications.

REFERENCES

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

B. Roe, H.-J. Y., et al. (2005). Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics*, 577-584.

Diener, H.-C., Bogousslavsky, J., Brass, L. M., Cimminiello, C., Csiba, L., Kaste, M., … others (2004). Aspirin and clopidogrel compared with clopidogrel alone after recent ischaemic stroke or transient ischaemic attack in high-risk patients (match): randomised, double-blind, placebo-controlled trial. *The Lancet*, *364*(9431), 331–

337.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple classifier systems* (pp. 1–15). Springer.

Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). *Fast subsequence matching in time-series databases* (Vol. 23) (No. 2). ACM.

Frank, P. M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results. *Automatica*, *26*(3), 459–474.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, *28*(2), 337–407.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Comput. Stat. Data Anal*, 367–378.

Gao, Z., Cecati, C., & Ding, S. X. (2015). A survey of fault diagnosis and fault-tolerant techniques-part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, *62*(6), 3757–3767.

Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., ... Liu, H. H. (1998). The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. In *Proceedings of the royal society of london a: Mathematical, physical and engineering sciences* (Vol. 454, pp. 903–995).

Isermann, R. (1984). Process fault detection based on modeling and estimation methods survey. *Automatica*, *20*(4), 387–404.

Isermann, R. (2005). Model-based fault-detection and diagnosis–status and applications. *Annual Reviews in control*, *29*(1), 71–85.

Isermann, R., & Ballé, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, *5*(5), 709–719.

Isermann, R., & Isermann, R. (2011). Supervision, fault-detection and diagnosis methods–a short introduction. *Fault-Diagnosis Applications: Model-Based Condition Monitoring: Actuators, Drives, Machinery, Plants, Sensors, and Fault-tolerant Systems*, 11–45.

J. Rosca, N. W. N. E., Z. Song. (2015). *PHM15 Challenge Competition and Data Set: Fault Prognostics, NASA Ames Prognostics Data Repository.* Retrieved from `http://ti.arc.nasa.gov/project/prognostic-data-repository`

Lawrence, R., Bunn, A., Powell, S., & Zambon, M. (2004). Classification of remotely sensed imagery using stochastic gradient boosting as a refinement of classification tree analysis. *Remote sensing of environment*, *90*(3), 331–336.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomforest. *R news*, *2*(3), 18–22.

Lindgren, J., & Rousu, J. (2002). Microscopy image analysis of bread using machine learning methods.

McCulloch, M., Cappo, M., Aumend, J., & Müller, W. (2005). Tracing the life history of individual barramundi using laser ablation mc-icp-ms sr-isotopic and sr/ba ratios in otoliths. *Marine and Freshwater Research*, *56*(5), 637–644.

Moisen, G. G., Freeman, E. A., Blackard, J. A., Frescino, T. S., Zimmermann, N. E., & Edwards, T. C. (2006). Predicting tree species presence and basal area in utah: a comparison of stochastic gradient boosting, generalized additive models, and tree-based methods. *ecological modelling*, *199*(2), 176–187.

PHM Society. (2015, June). *IEEE 2015 PHM Data Challenge.* Retrieved from `https://www.phmsociety.org/events/conference/phm/15/data-challenge`

Qin, S. J. (2009). Data-driven fault detection and diagnosis for complex industrial processes. In *Fault detection, supervision and safety of technical processes* (pp. 1115–1125).

Samy, I., Postlethwaite, I., & Gu, D.-W. (2011). Survey and application of sensor fault detection and isolation schemes. *Control Engineering Practice*, *19*(7), 658–674.

Schwabacher, M. (2005). A survey of data-driven prognostics. In *Proceedings of the aiaa infotech@ aerospace conference* (pp. 1–5).

Shepherd, K. D., Palm, C. A., Gachengo, C. N., & Vanlauwe, B. (2003). Rapid characterization of organic resource quality for soil and livestock management in tropical agroecosystems using near-infrared spectroscopy. *Agronomy Journal*, *95*(5), 1314–1322.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., & Feuston, B. P. (2003). Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, *43*(6), 1947–1958.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & chemical engineering*, *27*(3), 293–311.

Yin, S., Wang, G., & Karimi, H. R. (2014). Data-driven design of robust fault detection system for wind turbines. *Mechatronics*, *24*(4), 298–306.