Peer-reviewed version available at Sci 2020, 2, 75; doi:10.3390/sci2040075

Article

A Hybrid Approach: Dynamic Diagnostic Rules for Sensor Systems in Industry 4.0 Generated by Online Hyperparameter Tuned Random Forest

Ahlam Mallak [1,*], Madjid Fathi [1]

[1] Department of Electrical Engineering and Computer Science, Knowledge-based Systems and Knowledge Management, University of Siegen, Siegen, Germany; fathi@informatik.uni-siegen.de

* Correspondence: Ahlam.mallak@Ymail.com; Tel.: +49-1522-948-8004

Abstract: In this work, A hybrid component Fault Detection and Diagnosis (FDD) approach for industrial sensor systems is established and analyzed, to provide a hybrid schema that combines the advantages and eliminates the drawbacks of both model-based and data-driven methods of diagnosis. Moreover, spotting the light on a new utilization of Random Forest (RF) together with model-based diagnosis, beyond its ordinary data-driven application. RF is trained and hyperparameter tuned using 3-fold cross validation over a random grid of parameters using random search, to finally generate diagnostic graphs as the dynamic, data-driven part of this system. Followed by translating those graphs into model-based rules in the form of if-else statements, SQL queries or semantic queries such as SPARQL, in order to feed the dynamic rules into a structured model essential for further diagnosis. The RF hyperparameters are consistently updated online using the newly generated sensor data, in order to maintain the dynamicity and accuracy of the generated graphs and rules thereafter. The architecture of the proposed method is demonstrated in a comprehensive manner, as well as the dynamic rules extraction phase is applied using a case study on condition monitoring of a hydraulic test rig using time series multivariate sensor readings.

Keywords: industry4.0; fault detection; fault diagnosis; random forest; diagnostic graph; distributed diagnosis; model-based; data-driven; hybrid approach; hydraulic test rig

1. Introduction

In the last century, a new notion called "Industry 4.0" has emerged, in which the world has witnessed an industrial and technological revolution that caused the rapid spread of complex sensor systems in various domains and applications. These systems are literally everywhere; in aircrafts, transportation like cars and ships, computing systems such as computers, laptops, smart phones, embedded systems, and in many industrial applications such as factories, chemical reactors and nuclear power plants and many more countless examples. As long as these complex systems continue to function properly, they play a major role in providing help, comfort and assistance to our daily lives and they are even considered a necessity to the current structure of modern societies.

A fault in sensor systems within industry 4.0 can be defined as an unexpected event occurs at a certain point of time, which may trigger bigger events or a series of other unexpected events. Isermann and Balle [1] defined faults as:

An unauthorized, permitted or allowed deviation of one or more of the system's parameters, characteristics, behaviours or patterns from the normal or standard state of the system.

Based on the nature of complex industrial sensor systems mostly being non-linear, dynamic and having complex relationships between their components, it is highly complicated to predict faults in such systems. Faults consequences fall in a spectrum that ranges from harmless, ignorable faults to extremely disastrous ones that may lead to major economical and human catastrophes.

Most research suggests that faults in industrial sensor systems can be classified into three main categories; sensor faults, actuator and component/system faults [2].

- Component faults are the deviation of the system's characterized patterns and behaviours occurred to one, some or all parts/components of the system, other than the sensors and actuators. i.e. a fault in the cooling unit in an HVAC system, a fault in the transmission stick in an automobile or a fault in the CPU in a computing system.
- Actuator faults are the faults occur in actuation units and appears as a partial or complete malfunction of the actuation control. In other words, the actuators could be faulty when they fail to perform the actuation function i.e. stuck actuator. A complete fault in actuators can appear as a result of a burning wire, a cut, leakage, breakage or a presence of an actual physical object holding back the actuator preventing it from controlling the system's behaviour.
- Sensor faults are the faults represented by the sensors and their readings. Usually, these faults are noticed when the sensors are producing incorrect readings due to a physical fault in the sensor itself, broken wires or a malfunction in the communication channels between the sensors and the controlling unit, or the change of the sensor's reading could be an indicator (symptom) of a component or system fault.

Complex sensor systems within industry 4.0 contain hundreds of sensors attached in different locations of the system, wired or wireless, stationary or mobile, to continuously measure some key variables of these systems in real time. The data generated from sensors is considered a rich source of information from the analytical perspective. Since this type of data has a vast majority of unique patterns and worthwhile characteristics. Moreover, any sudden changes of these sensors' readings, or the appearance of any unexpected patterns that goes without notice, can lead to a major risk and serious consequences.

Fault Detection and Diagnosis (FDD) is the process of finding odd, extraordinary or unusual patterns in a given data, comparing to the patterns it usually forms in the regular or healthy state. These irregular patterns are most commonly called faults, anomalies or outliers [3]. In the last decade, FDD has been an interesting topic for many researchers applied in a wide range of applicational domains. Due to its enormous significance to provide the needed safety, security and reliability in many industrial systems. As well as, the vital role it plays in the fast detection of abnormalities and faulty patterns, which is essential in many industrial systems, especially the ones with harsh or highly restricted environments, systems that are prone to malicious attacks, sensor systems that contain fault-prone sensors or where the sensors' reading might be faulty or unusual. As a result, many FDD systems are developed for a specific domain, while others offer a more generic solution.

FDD can be classified into many groups and different types based on the strategy in which the diagnostic process is approached. The most common methodologies are model-based, data-driven and hybrid approaches [4,5] and [6].

- In model-based approaches, the diagnosis process follows a reference represented by a model, which can be in different forms and shapes. Such as a set of diagnostic rules, a semantic knowledge-base, a heuristic database, a simulation or an actual physical model. In such approaches, the quality of the diagnosis is completely associated with the accuracy of the created model.
- Data-driven approaches might not offer any knowledge of the physical system or a modelled simulation. However, they tend to extract data-driven knowledge by analysing the system recorded data and applying various methods to find hidden patterns and relationships that describe the unknown system and its behaviour. Although these approaches do not provide a good insight of the system and its processes, these approaches are extracted from the data (unlike model-based methods that are dependent on fixed rules and rigid knowledge) is considered a dynamic, general-domain solution.

The most famous approach related to data-driven FDD is using machine learning in a statistical manner or a black-box fashion. For statistical machine learning, there is a fair share of pattern recognition algorithms represented by various classifiers, regressors, and clusters, to solve numerous problems based on the availability of the class or label of the observations. In the black-box approach, shallow or deep neural networks can be applied to learn various diagnostic models based on the data. Data-driven diagnostic quality is dependent on various factors in which they are all related to the data. Such as the performance in data generation, data pre-processing, feature selection and/or feature extraction, and finally the chosen machine learning algorithm to solve the diagnostic problem.

- Hybrid-based approaches are the ones created by collaborating different approaches from the same group or different groups, all together to finally create a new diagnostic offspring approach that possesses shared qualities between the original parent approaches, and hopefully is good enough to eliminate one or more drawbacks of the parent approaches comparing to when applied individually. i.e. a combination of two data-driven approaches to form a new hybrid one or establishing a bridge between a specific method in data-driven and model-based to finally produce a hybrid offspring of shared qualities.

1.1. Challenges and Problem Statement

Model-based approaches for FDD tend to have a handful of disadvantages such as, the lack of dynamicity and generality, since they exhibit static knowledge for a specific domain stored in the model. The lack of or absence in handling sudden or novel fault occurrences (hence they are not pre-stored in the reference model), and the inability to automatically detect, fill or update the system gaps. The lack of credibility in knowledge acquisition because it is completely dependent on experts' reliability. And finally, the impossibility to learn from misdiagnosis and fault occurrence overtime.

Although, data-driven approaches might offer dynamic and general-domain diagnostic alternatives comparing to their model-based counterparts, they tend to have their share of challenges. i.e. the dependency on the data in case of poor data collection or tending to invalid sources, the dependency on possessing certain skills to apply data-driven processing and analysis methods, storing data necessary for learning and testing is resource and security expensive, and the additional expenses related to the needed supplementary hardware purchases and regular maintenance.

According to the mentioned drawbacks of each approach, it is essential to establish a hybrid approach that combines the positive sides of each one and eliminates as many as possible of their limitations.

This work is intended as an extension to the method in [7]. The algorithm in [7] demonstrates a model-based component FDD method based on using diagnostic graphs created by static/constant diagnostic rules extracted from semantic ontology. In other words, the system model represented by the Ontology [8] is fed directly with the expert knowledge, and later used to generate diagnostic graphs that links between various symptoms and their faults. The created graphs using the model-based approach alone are lacking the dynamicity and the generality, where they are only applicable to a certain system or model that they were created upon. Thus, a more general and dynamic approach is needed.

Creating dynamic diagnostic graphs using data-driven approaches such as Random Forest (RF) can be beneficial. However, because of their dynamic nature, these models are hard to use in structured or distributed systems without following some guidelines, graphs or clear steps. Furthermore, data-driven approaches require more time and resources to process and store the needed data. Thus, a strong necessity to create a general domain, dynamic but structured enough algorithm, to guarantee general-domain application. Additionally, a decrease in the time and resource complexity constraints required by online data-driven approaches.

1.2. Our Contribution

On one hand, this work demonstrates a unique architecture to deploy RF in FDD beyond its ordinary application as a data-driven methodology. Usually, RF is used as a classifier, feature selection, and when certain adaptations are made, RF can also be used for unsupervised learning. However, the literature is lacking the use of RF for model-based FDD or hybrid approaches beyond the data-driven combination ones.

On the other hand, the architecture and application of an algorithm to diagnose component faults for industrial sensor systems, in a dynamic and distributed fashion is introduced. Where a hybrid approach between model-based and data-driven methods is established.

Finally, in this work, a development and extension of the work in [7] is proposed, by offering a dynamic and general domain approach, with the possibility of deployment in distributed systems. Represented in generating dynamic diagnostic graphs using RF.

It is worth emphasizing the reasons behind choosing RF to contribute as part of this work, instead of many other machine learning classifiers. RF has been chosen in this work due to its high accuracy for fault classification when applied to real-life dataset and compared to a wide variety of other classifiers of different functionalities and workflow. This comparison is explained in detail under the results section. Furthermore, for the creation of diagnostic graphs, it is easier to incorporate a classifier that provides some sort of directed graph structure in its flow, which is supported by the tree structure of RFs.

The rest of the work is organized as follows: Section 2 shows RF related work for FDD in industry 4.0. In section 3, some literature of RF is introduced. Section 4 provides the materials and methods used in this work. Section 5 demonstrates the system overview and experimental results. Finally, section 6 represents the discussion and future work.

2. Related Work: RF for FDD in Industry 4.0

For the last few decades, RF has been used widely to perform FDD and monitoring applied in various fields and applications, such as in industry 4.0. The literature demonstrates several techniques to apply RF for the purpose of outlier detection, either exclusively or incorporated with other algorithms to form some sort of a hybrid approach aimed to fulfil an intended research or applicational purposes.

The most common methodology of deploying RF is as a classifier. RF is intended to achieve an optimized, supervised and structured resolution for labelled problems, which is proven to have more accurate results compared to many other supervised machine learning algorithms. In [9] RF is compared to numerous classifiers of different functionality to overcome two occurring sensor faults in Wireless Sensor Networks (WSNs), which are spike fault and data loss fault. This study represents an elaborated comparison between RF, Support Vector Machine (SVM), Stochastic Gradient Descent (SGD), Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Probabilistic Neural Network (PNN). Using Detection Accuracy (DA), Matthews Correlation Coefficients (MCC), True Positive Rate (TPR), and F1-score as the comparison criteria that determine the overall rank of each method. As a result, RF is proven to have the highest rank of all the above classifiers in WSN's sensor fault classification. In addition, another study in [10] showed similar results in proving the superiority in performance of RF in the field of WSN, but this time while detecting four different sensor faults; gain, offset, constant and out of range faults.

In [11], another example of using RF in a solo fashion to achieve FDD in industrial sensor systems applied to unmanned aircraft vehicle. This study deployed a brilliant interpretation of RF and feature

importance, to extract a weighted similarity metric based on the data priority represented by RF. The induced similarity measure is then used to perform FDD.

RF can also be used combined to different approaches instead of using directly as a classifier to achieve FDD in industrial systems. Usually, any hybrid approach is originated to optimize the individual forming methods combined, or to establish a customized solution that fulfils additional system goals or requirements.

In [12], a hybrid approach is established to detect faults of rolling bearings, which if left undetected can lead to major consequences in the performance of the rotating machine. This hybrid approach combines Wavelet Packet Decomposition (WPD) method to extract new enhanced features from the bearing vibration signal provided from *n* number of sensors, using signal-to-noise ratio and Mean Square Error (MSE). Followed by the step of mutual, dimensionless index construction, which will be fed to the fault database and contribute as the data necessary to train and test the RF model.

Moreover, another example for a hybrid FDD approach using RF is [13], This method demonstrates the effect of combining genetic algorithm and RF to increase the classification accuracy of the FDD process of an induction motor.

In this recent work [14], RF is used in a hybrid fashion with Feedforward Neural Network (FNN) to investigate the relationship(s) among multi-modal signals, extracted from electrochemiluminescence (ECL) sensor located in a smartphone and the concentration of $Ru(bpy)_3^{2+}$ luminophore and its electrochemical data. Establishing such correlation is essential for building optimized and cheaper diagnostic devices. Understanding the hidden relationships between each modality may lead to creating diagnostic rules, which can be used for FDD in later stages. Thus, this study is included with the application of RF in FDD related work.

Beyond the intensive use of RF in industrial sensor systems, RF can be used in a smaller range, for many reasons and purposes exceeding the industry. One of the common applications of RF is in the medical field using sensing modalities. In [15] a recent study shows an application of RF to reduce the fallacious clinical alarms. i.e. the Arrhythmia alarms. In case of false Arrhythmia alarms occurrences, that may lead to elevation in the patient and staff stress level, as well as causing unnecessary pressure on the intensive care staff. According to the study, the application of RF detects the true from the faulty calls has significantly reduced the amount of false calls concerning five main types of arrhythmia.

3. RF Literature Review

Random Forests (RF), also known as Random Decision Forests (RDF) are a machine learning algorithm of an ensemble nature that is originated for classification and regression purposes. RF is an ensemble learning algorithm of decision trees methodology which is intended to optimize decision trees and resolve their tendency to form overfitting patterns over the training dataset.

In order to understand the mathematics behind RF, it is highly recommended to go through the explanation of decision trees and how they work in the first place.

3.1 Decision Tree

Decision trees in data mining are a commonly used supervised technique to solve classification and regression problems. Where a set of observations and their labels or classes are already known and used to make various predictions [16]. In data mining, decision tree algorithm is divided into two main types; classification and regression trees. In 1984, Breiman et al [17] combined the two types together under the same category using the term Classification And Regression Tree (CART).

Decision trees are called this way because they are visualized in a tree structure, in which is created by recursively splitting the training dataset from top to bottom, forming the first level node of depth zero called the “root”, followed by going down the tree forming higher depths and continuously splitting into successor children nodes. The splitting process is determined using different rules that determine the impurity of a certain node, and upon the selection of the splitting criteria [18].

Table 1, represents some of the most common node impurity criteria based on the tree type, linked to their scientific formula, and a brief description of the splitting mechanism.

In this work, we will only focus on classification trees. Thus, only classification trees splitting criteria is discussed and deeply explained. For more information check [19].

Table 1. Node Impurity Splitting Criteria

Tree Type	Criterion	Mathematical Formula	Description
Classification Tree	Gini Impurity	$\sum_{i=1}^{C} f_i(1-f_i)$	f_i is the frequency of the class i. Where C is the total number of classes/labels
	Entropy	$\sum_{i=1}^{C} -f_i \log(f_i)$	

Gini Impurity is one of the most common metrics used to determine the best split for classification trees. Gini impurity works by finding the probability for each class/label in which they were incorrectly classified. In other words, Gini impurity is the probability of the falsely classified subjects in respect of a randomly chosen split point based on the original distribution of the dataset. During the decision tree training process, the best split is represented by the split that maximises the Gini gain. Where the Gini gain is calculated by subtracting weighted impurities of the left and right branches of the chosen split from the original impurity existing in the whole dataset, before randomly choosing the splitting point.

3.2 From Bootstrap Aggregation(Bagging) To RF

Furthermore, there are some algorithms classified under ensemble learning category that allow the possibility of creating multiple different trees over the same dataset, to contribute in minimizing the over-fitting problem decision trees usually suffer from, especially when the sample size provided is relatively small. The two main types of such ensemble methods are boosted ensemble trees [20] [21] and bootstrap aggregated or bagged trees [20] [22].

Boosted trees are a sequential type of ensemble decision trees, where the optimal shape of the tree is established incrementally by adjusting the tree continuously based on the arrival of new instances. The most famous boosted trees algorithm is AdaBoost method.

Bootstrap aggregated or bagged trees are a parallel type of ensemble decision trees, generates multiple numbers of decision trees concurrently, by resampling the training dataset with replacement. The final prediction for such methods is made by voting the results of the created trees altogether. What is worth to be mentioned is that random forests are an example of a Bootstrap aggregating methods to optimize the traditional decision trees methodology [22].

Random forests are an optimization algorithm of decision trees, under the ensemble learning sub-category, intended to perform different tasks such as, classification, regression and many others. The core of this algorithm relies on creating multitude of parallel decision trees based on dividing the feature space each time and deploying the chosen sub-space to form the tree of choice. The prediction decision of random forests is made by majority vote of all the separately created trees. Random forests started as “Stochastic discrimination” approach created by Eugene Kleinberg [8]. Which was inspired by the formula created by Tin Kam Ho [23] to deploy the understanding of random subspaces and how to use them in a practical approach. Recently, random forest algorithm is trademarked by Leo Breiman and Adele Cutler owned by Minitab, Inc in 2019 [24]. The registered algorithm represents an extension of the formula introduced by Ho [23] and the “Bagging” idea created by Breiman [22] and [17].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 July 2020 doi:10.20944/preprints202007.0548.v1

Peer-reviewed version available at Sci 2020, 2, 75; doi:10.3390/sci2040075

Bagging algorithm whole idea depends on randomly choosing a subset of the original training set with placement, to perform an S number of classification or regression tasks, to finally make the overall decision of the performed task using all the learners created.

Generally, the trees created by the bagging algorithm alone tend to be highly correlated and, in most cases, the same tree is being generated for multiple of times. Due to, simply, training multiple trees over the same dataset with placement that can easily generate highly correlation between the formed estimators. The best way to introduce some sort of de-correlation between the trained trees is by feeding the algorithms different datasets. A new dataset can be formed from the original dataset by using the random subspace algorithm [25] to not only randomly choose the data points, but also concurrently pick randomly a feature from the feature space, to act as a new splitting point. Random forests use random subspace method to de-correlate the trees formed using the bagging method alone.

Random subspace algorithm is highly identical to bootstrap aggregation in many ways. The only difference is that in random subspace the features are the subject of bagging and they are considered as the "Predictors" or "Random variables" that would be sampled with replacement to create predictions for each learner. Thus, random subspace is also known as attribute bagging [26] or feature bagging. Random forest algorithm is a combination of bootstrap aggregation to sample the training dataset, and random subspace algorithm necessary to sample the features, to create splitting points that results in generating multiple estimators with high level of distinction and accuracy.
Algorithm 2 shows a detailed explanation of random forests.

Algorithm 2 Random Forests

1. Given a training labelled dataset

$$(\boldsymbol{X},\boldsymbol{Y}) = \{(\boldsymbol{x_1},\boldsymbol{y_1}),(\boldsymbol{x_2},\boldsymbol{y_2}),(\boldsymbol{x_3},\boldsymbol{y_3}),\dots\ (\boldsymbol{x_n},\boldsymbol{y_n})\}$$

   Where X is a set of data and Y is their corresponding response or label
2. For S number of times. Randomly chose N number of random samples with Replacement from $(\boldsymbol{X},\boldsymbol{Y})$. and M number of features.

   for s=1,2,…S:

   a) (Bootstrap Aggregation):

   With Replacement Randomly choose N training samples called Xs, Ys from (X,Y)

   b) (Random Subspace):

   Randomly Choose M number of features from the feature space

   c) Choose the best split among the features randomly selected in b)

   d) Grow the random forest tree Ts to the data chosen in a) using the best split in
3. Ensemble Algorithm Output in a form of S number of trees $\{\boldsymbol{T_s}\}_1^S$
4. To make predictions of a new test point x

   - Regression Predictions:

   The average of all predictions from the S experiment

$$\hat{\boldsymbol{f}} = \frac{\boldsymbol{1}}{\boldsymbol{S}}\sum_{\boldsymbol{s=1}}^{\boldsymbol{S}} \boldsymbol{f_s}(\boldsymbol{x})$$

   -Classification Predictions:

   Use majority voting to find the predicted class.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 July 2020 doi:10.20944/preprints202007.0548.v1

Peer-reviewed version available at Sci 2020, 2, 75; doi:10.3390/sci2040075

4. Materials and Methods

4.1 Condition Monitoring Of Hydraulic Test Rig Data Set

This dataset [27] represents real measurements of multivariate, time-series sensors, placed in a hydraulic test rig. The purpose intended for the data collection is to monitor and assess the hydraulic system health condition.

The outcome of this experiment yielded a success of collecting sensor data of various system health degrees of different components of the hydraulic system, such as the cooler, valve, pump and accumulator.

The system consists of six pressure, four temperature, two volume sensors and one vibration sensor which all possess a constant cycle of 60 seconds. Each cycle the sensors are collected, while the condition of the four main hydraulic components; cooler, valve, pump and accumulator are monitored and observed. The component health ranges from completely healthy to totally damaged, and each condition degree is decoded into a numerical value to facilitate the application of statistical and data mining approaches.

This dataset has been used by many researchers to perform sensor fault monitoring. i.e. constant, shift, bias and peak. [2]. Moreover, this dataset is also beneficial to perform component or system FDD such as the application researched in [29]. As well as, being applicable for creating and testing feature extraction and selection algorithms in [30].

5. Results

5.1. System Model Overview

The proposed system scenario represents a possible method to merge random forest as a data-driven approach for industrial FDD and model-based FDD approaches into a final hybrid approach, which possesses the powerful features of both approaches. This technique eliminates the main drawbacks of each approach individually such as, the lack of dynamicity and response to sudden occurrences in case of traditional model-based FDD. As well as, providing validated, accurate and dynamic diagnostic rules that contribute massively in reducing the diagnostic time and computational needed resources, comparing to their online data-driven counterparts. Figure 1, shows the two main diagnosis phases used in this research; data-driven and model-based, and how these two methods are combined into a new improved approach.

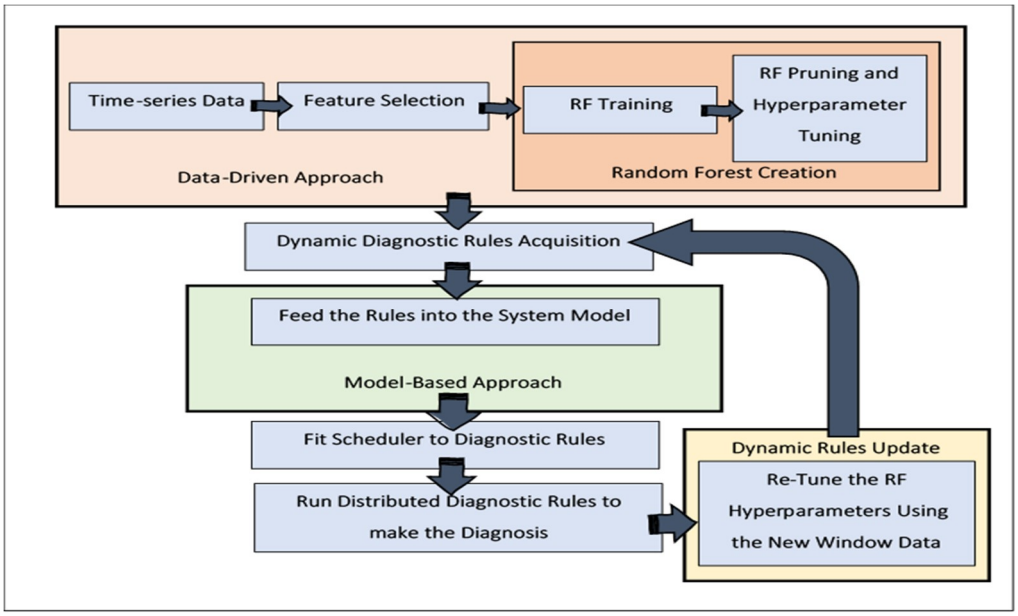


Figure 1. Hybrid-based FDD System Overview

The following is a comprehensive explanation of each phase:

- Data- Driven Phase:

This phase consists of multiple internal steps essential to learn the best possible dynamic diagnostic rules using random forest algorithm. Below, each step is discussed in an elaborate manner:

- Multivariate Time-Series Dataset

In this work, the dataset used is a multivariate, time-series dataset, of six pressure, four temperature, two volume sensors and one vibration sensor which all possess a constant cycle of 60 seconds, placed in a hydraulic test rig to monitor its condition over time. For more details about this dataset and its previous applications, please refer the data collection and generation section, in this work.

- Feature Selection

Complex sensor industrial systems often have hundreds or even thousands of sensors connected, simultaneously transmitting sensors' reading data crucial to monitor and control those systems. Each of which is considered a feature for analysis and model training. Thus, creating diagnostic models that only include valuable features is a necessity.

Implementing a model with less but more meaningful features has a significant impact on the overall system. First, the diagnostic model become simpler to analyze and interpret when fewer elements are included. Second, by eliminating some features of the dataset, the data would be less scattered hence less variant, which can lead to reducing overfitting. Finally, the main reason behind feature selection is generally to reduce the time and computational costs required to train the model.

In practice, RF algorithm can be applied to carry out feature selection as well. Simply because the features are implicitly ranked based on their impurity during the formation of each decision-tree creating the forest. In other words, when top-down traversing a tree in RF, the nodes toward the top happened to have the largest impurity decrease metric, also known as Gini Impurity, comparing to the nodes at the bottom. Thus, by determining a particular impurity decrease threshold, it is possible to prune the tree below this tolerance, in order to establish a subset of the most fitting or important features.

The data-driven FDD method implemented in this work is RF. In intention of reducing the computational cost as possible, RF is also used to perform feature selection using what is known as feature importance or permutation importance [22] and [31]. Since, Gini impurity calculations are already measured during the RF training process, and only a slightly bit of additional computations are required to complete the feature selection process.

In the context of RF, Mean Decrease Accuracy (MDA) or permutation importance or feature importance of a variable X_n to predict Y of classes is computed by the summation of the Gini impurities of X_n for all the nodes d where X_n is present and used. Followed by, the mean of the impurity decrease metric of all the trees D in the forest. The following equation comprehend the concept of feature importance using RF.

$$Importance(X_n) = \frac{1}{no.Trees} \sum_{no.Trees} \sum_{d \in D: v(s_d)=X_n} Gini\_Impurity(X_n), \quad (1)$$

Where X_n is the feature of interest. $v(s_d)$ is the feature/variable used to split s_d.

The most popular implementation of feature importance provided by RF, is the Python library Scikit-learn where a pre-defined function feature_importances_ is directly executed given the learned RF model. However, a team of data scientists at the University of San Francisco pointed out some bugs associated to this function and implemented an alternative to generate more accurate feature importance results in [32].

- Hyper-Parameter Optimization

The foremost goal of any machine learning algorithm is to minimize the expected loss as possible. To achieve this, it is consequential to deploy some optimization equations to select the optimal values for some, or all the hyperparameters of the machine learning algorithm of focus.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 July 2020 doi:10.20944/preprints202007.0548.v1

Peer-reviewed version available at Sci 2020, 2, 75; doi:10.3390/sci2040075

RF algorithm has plenty of hyperparameters. On one hand, some are implemented on the overall forest level such as, the number of subjects randomly drawn from the dataset to form each tree, the choice of with or without replacement regarding the samples selection and most importantly is the number of trees in the RF. On the other hand, some hyperparameters are on a tree level, which control the shape of each tree in RF. i.e, the number of features drawn for each split, the selection of splitting rules, the depth of each tree and many others. These parameters are typically selected by the user. Consequently, creating a method to efficiently select these hyperparameters can influence the performance versus the cost of RF significantly. In addition, the recent research done in [33] emphasizes the significance of hyperparameter optimization specifically for RF parameters, as well as providing deep comparisons between numerous tuning and optimization mechanisms and software.

One of the key tuning strategies for RF, is using searching algorithms to look for optimal parameters in a pool or grid of selected ones. Search techniques differ in their way of pool or grid creation, based on the mechanism applied to choose the successful candidates forming the bag of options. Some searching strategies use all the possibilities available as candidates to be exhaustively investigated, one by one to select the optimal choice, as in grid search algorithm. However, in random search, the bag of candidates are drawn randomly from the overall existing possibilities, which is not only a precious asset for reducing the search complexity, but also studies have proved that random search produces better accuracy scores for parameters optimization comparing to grid search [34].

Random search refers to a group of searching algorithms that rely on randomness or pseudo-random generators as the core of their function. This method is also known as a Monte Carlo, a stochastic or metaheuristic algorithms. Random search is beneficial for various global optimization problems, structured or ill-structured over discrete or continuous variables [35].

Below is the pseudo code describing the workflow of a typical random search algorithm.

Algorithm 3 Random Search Algorithm

Let RF is the cost function to be optimized or minimized. C is a candidate solution in the search-space R^n.

1. Select termination condition TC. i.e. specific fitness measure achieved, or max number of iterations reached, and so on.
2. Initialize C. $C = Random\ position \in R^n$
3. $For\ TC$
   a) Randomly choose another position C_{new} from the radius surrounding C (the radius of the hypersphere surrounding *C)*
   b) if $\mathrm{RF}(\mathrm{C_{new}}) < \mathrm{RF}(\mathrm{C})$ then
      $C = C_{new}$

In practice, Scikit-learn library for Python machine learning provides a method; RandomizedSearchCV which can be provoked by creating a range for each hyperparameters subject of optimization. By calling RandomizedSearchCV method over the predefined range, random search is performed to randomly select a candidate grid of possibilities within the range, then applying K-fold cross validation technique over the created grid. For additional examples about this method, refer to [36].

- Model-based Phase

This phase represents a clear model of the system, whether it is an actual physical model, a simulation, a knowledge-base semantically connecting the system component together, or a relational database. Based on the system model nature, the extracted nested, conditional rules from the random forest are transmitted to a suitable form. i.e. in knowledge-based systems such as ontologies, the rules are converted into SPARQL [37] semantic queries, a regular SQL queries in case

the system model is represented by a relational database, or in a simpler fashion use the extracted rules as a small conditional code, which runs every diagnostic window to perform the diagnosis. This phase is crucial to minimize the online diagnostic time and computational power needed, comparing to provoking the testing RF algorithm over and over for each sliding window. Moreover, it provides a possibility to introduce online distributed mechanisms given the rules and the graphs creating them.

- Dynamic Rules Update Phase

In this phase, the new time-series data generated by the system for a certain amount of sliding windows, is stored and used to update the originally created RF, by performing the hyperparameter tuning again to find out if any alteration of the RF parameters could reduce the size of the overall RF and increase the accuracy at the same time. The new updates selection or rejection decision is highly dependent on the accuracy of the newly tuned RF.

5.2 Experimental Results

In this experiment, RF is used following the steps in the data-driven flowchart in Figure 1, to generate dynamic diagnostic rules to diagnose and monitor the health of a hydraulic test rig. Provided in the dataset, each component condition ranges between full efficiency, reduced efficiency and close to total failure. In this experiment, for the sake of simplicity, the healthy state is represented by the full efficient cycles, and the failures are represented by the cycles where the component is close to failure, while the partial failure state is being excluded. Based on the previous fault description, there are four types of total failure in four different components to monitor; cooler total failure state, valve close to total failure, internal pump has a severe leakage and hydraulic accumulator close to total failure. Table 2, explains the definition of each fault chosen for this experiment and some example cycles that contains each fault.

Table 2. Hydraulic Test Rig Chosen Faults and Their Full Description

Status	Status Description	Example Cycle no.
Healthy	All components are healthy and in full efficiency mode.	1788, 1789,1790
Cooler Fault	Cooler has a total failure, and the rest of the components are fully efficient.	1,2,3
Valve Fault	Valve close to total failure, and the rest of the components are fully efficient.	1759, 1760, 1761
Pump Fault	Internal pump has severe leakage, and the rest of the components are fully efficient.	1675, 1706, 1707

Accumulator Fault	Hydraulic Accumulator close to total failure, and the rest of the components are fully efficient.	1465, 1466, 1467

The hydraulic system described in this experiment contains eleven sensor readings of three types of sensors located in different components of the hydraulic test rig. Six Pressure sensors labelled as *PS1*, *PS2* up to *PS6*, four temperature sensors *TS1-TS4*, and finally one vibrational sensor labelled as *VS1*. The readings of all the eleven sensors from various cycles containing the five different statuses shown in the table above, are collected in one labelled dataset of eleven features necessary to perform RF training and analysis.

As mentioned earlier, the selection of RF as the classification method in this work is done after carefully comparing the results of RF in respect of other famous classifiers such as, Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbour (KNN), regular decision tree (CART), Naïve Bayesian (NB) and Support Vector Machines (SVM). The supervised machine learning methods shown above along with RF are used to perform a multi-class classification task, to classify the hydraulic test rig faults described in table 2. The following table, table 3 shows the classification results after performing multi-class classification using different classifiers. It is demonstrated clearly that CART and RF has elevated accuracy comparing to the rest of the approaches. However, RF is an optimization of CART which overcomes its tendency to form overfitted relationships with the training dataset.

Table 3. RF Accuracy Results Comparison to Some Other Classifiers for Hydraulic Test Rig Fault Classification

Method	**LR**	**LDA**	**KNN**	**CART**	**NB**	**SVM**	**RF**
Accuracy	0.780469	0.778646	0.932031	**0.989844**	0.704167	0.923177	**0.985198**

Non-zero feature importance method is used to neglect the features with less impact of the learning process, by concentrating only on the features that contribute more to the model accuracy. The table below shows the importance of each sensor to the RF model calculated using equation 1.

Table 4, shows the calculated importance for each one of the eleven sensors. There are variety of options of which these importance values are analysed and evaluated to achieve feature selection. One can pick the highest importance feature alone to represent the whole features, or the highest three, highest six or just the non-zero ones to represent the whole pack. However, the most convincing approach is by experimenting all the possibilities and making a logical accuracy versus complexity trade off. For each selected feature(s) scenario, the RF accuracies and the time complexity given $O(T \log n)$ equation are calculated. Where T is the number of trees in the RF and n is the size of the input data used for training. Assuming that the number of trees T is constant for all the feature trials. Hence, the time complexity is a factor of input data size represented by the number of features included without sacrificing much or none of the model accuracy.

Table 4. Feature Importance Calculated for Each Sensor Feature

Sensor label	PS1	PS2	PS3	PS4	PS5	PS6	TS1	TS2	TS3	TS4	VS1
Importance X100 (%)	0	0	0	29.689	16.036	14.118	6.446	8.977	8.920	9.509	2.875

In Table 5, four different RF model training experiments are conducted to find out the best number of features required to train an RF on a hundred decision trees. In the first trial, the most important feature *PS4* is used alone to train the RF model. The second trial used the top three important features *PS4*, *PS5* and *PS6*. The third trial applied the highest six features. Finally, only the non-zero features are selected to train the RF model. For all the above four experiments, the random forest has fixed hyperparameters which were randomly chosen; one hundred trees and the maximum depth of five. Furthermore, 10-fold cross validation technique is used to compute each trial's accuracy.

Table 5. RF Accuracies Using Different Features Based on Their Importance

Trial Description	No. features	RF Accuracy (%)
No feature selection	11	0.985
Highest Feature	1	0.707
Highest Three	3	0.977
Highest Six	6	0.986
Non-zero importance	8	0.981

For this experiment, the highest six features are used for the training process since these features provided the best accuracy among all trials and shown lower time and space accuracy comparing to the training using eleven and eight features, respectively.

The following figure, Figure 2, shows how the time complexity $O(T \log n)$ and space complexity $O(n)$ for the RF are directly proportional to the number of features used. It is crucial to emphasize that the amount of accuracy sacrificed, and the added complexity tolerances are totally dependent on the system used and one's preferences. i.e. some other researchers would use the highest three features with 0.977 accuracy, if ones are willing to lose more accuracy in expense of the dramatic drop in both time and space complexities.

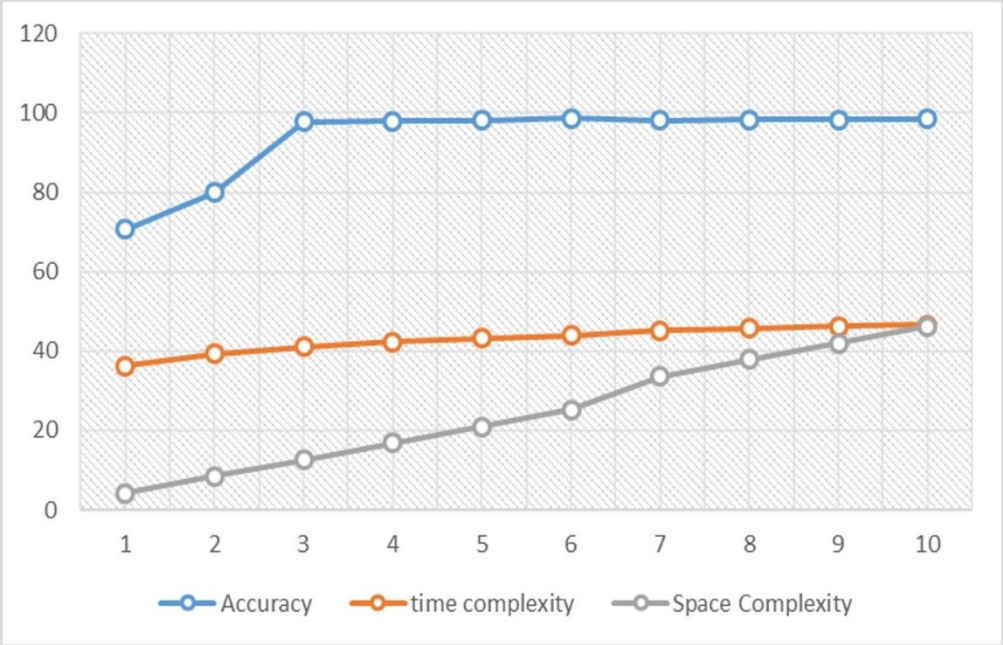


Figure 2. RF Accuracy, Time and Space Complexities in Respect of Number of Features

The next step is tuning the hyperparameters of the RF applied on the dimensionally reduced dataset of the selected six features; *PS4*, *PS5*, *PS6*, *TS2*, *TS3* and *TS4*. The hyperparameters subject of tuning in this experiment are the number of trees in the RF, the maximum depth of the tree, minimum number of samples required to split a node and minimum number of samples required to form a leaf node. As the purpose behind RF creation in this research is to establish a set of base rules for fault diagnosis, the main hyperparameter of focus is the number of trees to ensure lessening the complexity as much as possible, as well as, minimizing each tree's depth if possible.

A random grid of hyperparameters is created by applying random search over a pre-defined range for each parameter separately. i.e. the number of trees is pre-defined to range between one and one thousand, and only a hundred possibility is selected from the range to form the random grid for this hyperparameter. Followed by RF training using one of the randomly selected pair of features at a time. The selection is validated using 3-fold cross validation to calculate the accuracy of the RF model over a particular set of hyperparameters. A hundred set of randomly selected parameters are used to create the grid, which means 300 RF model training has been successfully executed considering 3-fold cross validation over the hundred set of possibilities in the grid. Finally, the set of hyperparameters with the highest accuracy when applying 3-fold cross validation is the one selected to generate the diagnostic rules.

The RF model trained after applying feature selection with randomly chosen parameters yielded an accuracy of 0.9865% using a hundred decision trees forming the RF with maximum depth of five. However, the best hyperparameter tuned using cross validation over random search grid improved the accuracy with 0.32% to reach 0.9865% . Additionally, using only 49 trees in total instead of one hundred over the same depth, which has dramatically decreased the complexity and size of the generated RF rules while increasing the accuracy and speed of the diagnosis.

The best hyperparameters selected from the grid has 49 number of estimators, minimum sample split of two, minimum leaf samples of one and maximum depth of 83. It is worth to be mentioned that the accuracy of the RF using all the best hyperparameters increased the accuracy to 0.99%, but this slight rise in the accuracy is not worth it, especially when it is compared to the massive increase in the time and size of each tree in the RF, due to the large increase in the maximum depth.

Figure 3, shows the one of the decision trees in the RF after feature selection and hyperparameter tuning.

Each tree in the RF can be translated into a set of nested if-else statements of rules. Moreover, the formed dynamic rules from the RF can be fed inside various system models to generate a hybrid approach out of the data-driven and the model-based ones. The dynamic rules extracted from the RF can be used as they are, converted to SQL queries if the model is a relational database or a SPARQL queries if the system model is represented by a semantic knowledge-base such as ontologies.

The diagnostic rules can be extracted from the RF dynamically, using a few lines of code in Python language. For the sake of simplicity, Figure 4, shows the tree in Figure 3, pruned in a way that only the positive part of the condition after the root node is remaining, connected to a series of nested if statements showing how this part of the tree is translated into clear dynamic rules.

The work in [7] provided a graph-based FDD system for industrial systems using a model-based approach, based on creating a knowledge-base of the system subject of diagnosis, such as ontologies. Followed by manually feeding a set of static diagnostic rules created by the system expert into the ontology, in a way that forms a causation relationship between the system sensors and the faults they lead to. In this work, we propose creating dynamic rules using RF, extract these rules and feed them in the ontology instead of the expert rules that are static, unreliable, and unverifiable. Furthermore, the extracted diagnostic rules using RF can be applied in a variety of forms to fit the model expressing the system.

Figure 5, showcases the extracted rules from the optimized, hyperparameter tuned RF, and how these rules are transformed to various forms and types to match the nature of the system model. As mentioned before in this chapter, the diagnostic rules can be translated into SQL queries in case a relational database is the system model, or SPARQL queries if a semantic knowledge-base, such as ontologies are used to represent the system. The rules extracted from each tree may be scheduled separately, or all together with the trees forming the RF.

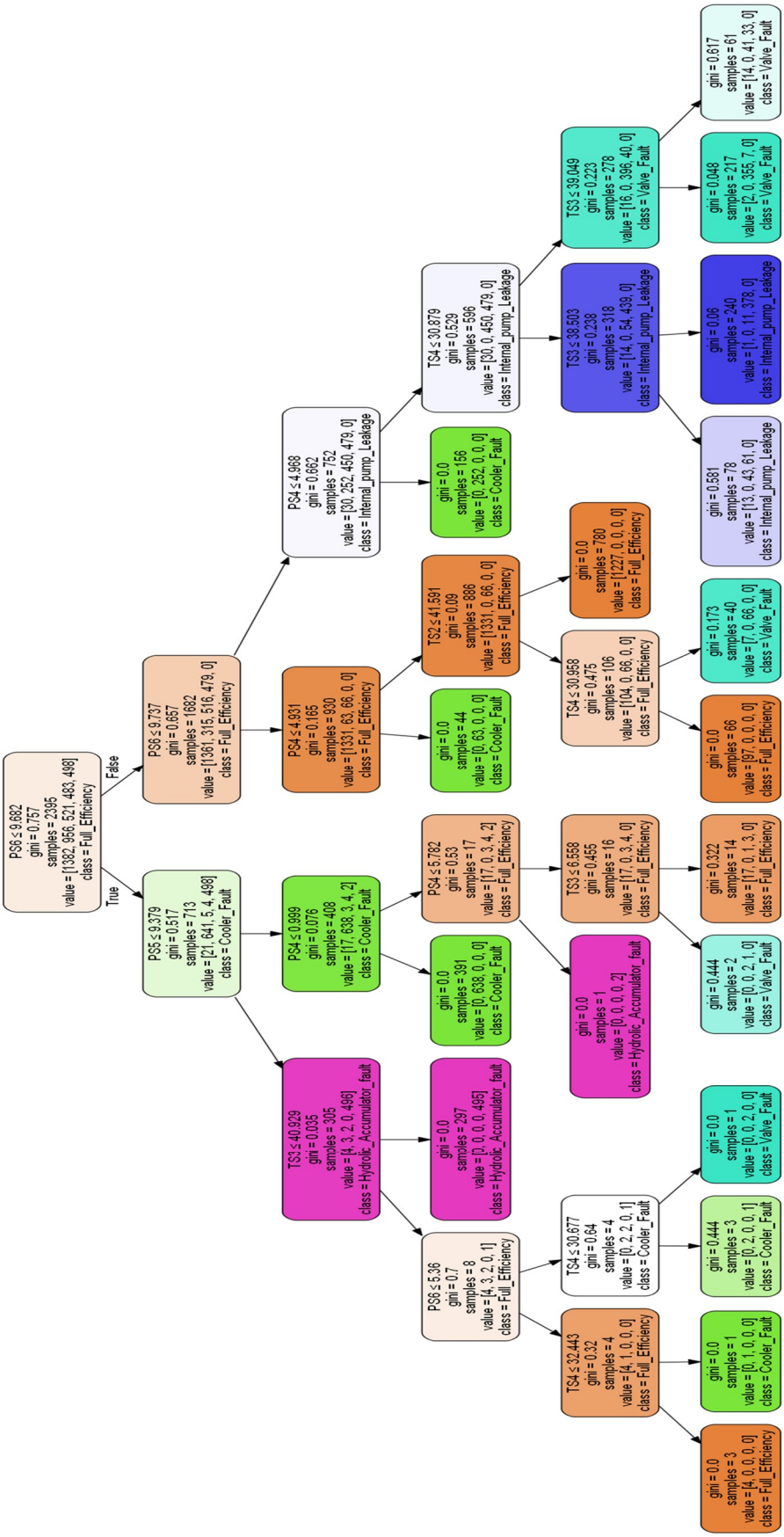
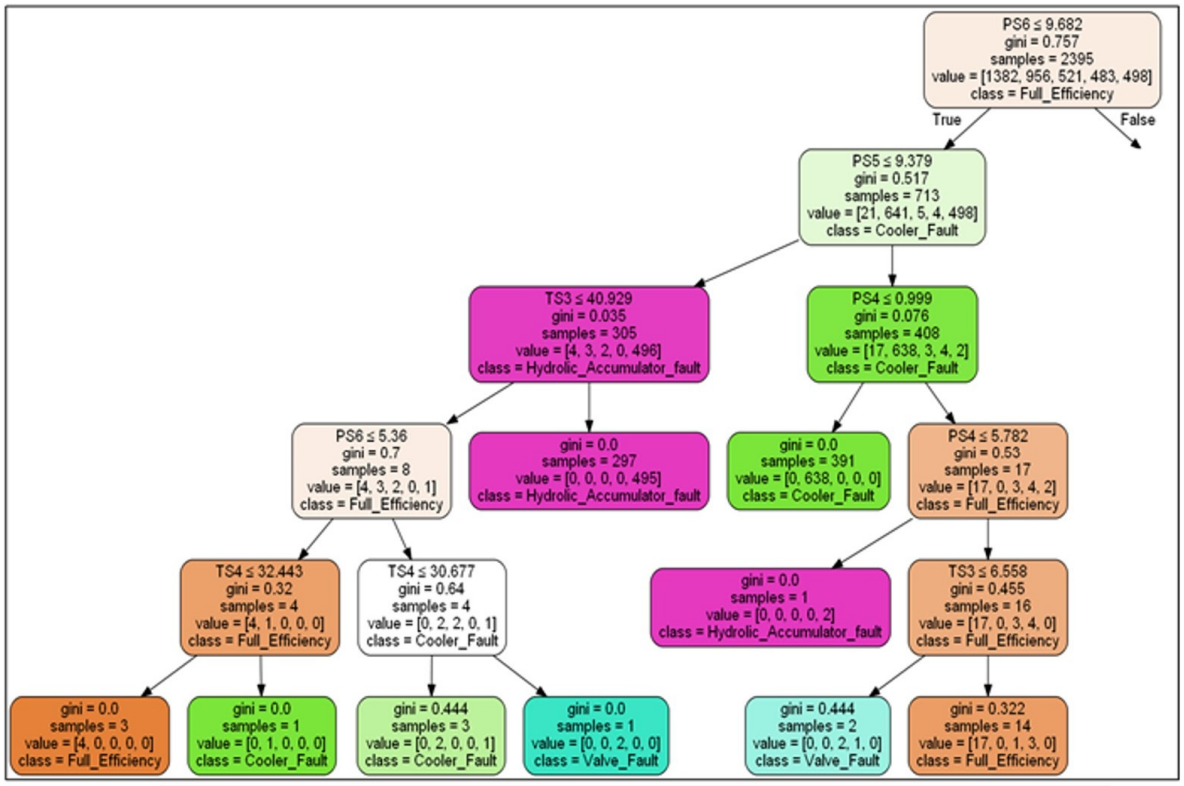


Figure 3. One of the Decision Trees in the RF after Feature Selection and Hyperparameter Tuning

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 July 2020 doi:10.20944/preprints202007.0548.v1

Peer-reviewed version available at Sci 2020, 2, 75; doi:10.3390/sci2040075




```
if PS6 <= 9.682436466217041:
  if PS5 <= 9.378981590270996:
    if TS3 <= 40.929338455200195:
      if PS6 <= 5.359611511230469:
        if TS4 <= 32.44318962097168:
          return [[4. 0. 0. 0. 0.]]
        else:  # if TS4 > 32.44318962097168
          return [[0. 1. 0. 0. 0.]]
      else:  # if PS6 > 5.359611511230469
        if TS4 <= 30.676508903503418:
            return [[0. 2. 0. 0. 1.]]
        else:  # if TS4 > 30.676508903503418
          return [[0. 0. 2. 0. 0.]]
    else:  # if TS3 > 40.929338455200195
      return [[0.   0.   0.   0. 495.]]
  else:  # if PS5 > 9.378981590270996
    if PS4 <= 0.9992818832397461:
      return [[0. 638.   0.   0.   0.]]
    else:  # if PS4 > 0.9992818832397461
      if PS4 <= 5.781721115112305:
        return [[0. 0. 0. 0. 2.]]
```

Figure 4. Diagnostic Rules Extraction from Parts of a Tree in the RF.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 July 2020 doi:10.20944/preprints202007.0548.v1

Peer-reviewed version available at Sci 2020, 2, 75; doi:10.3390/sci2040075

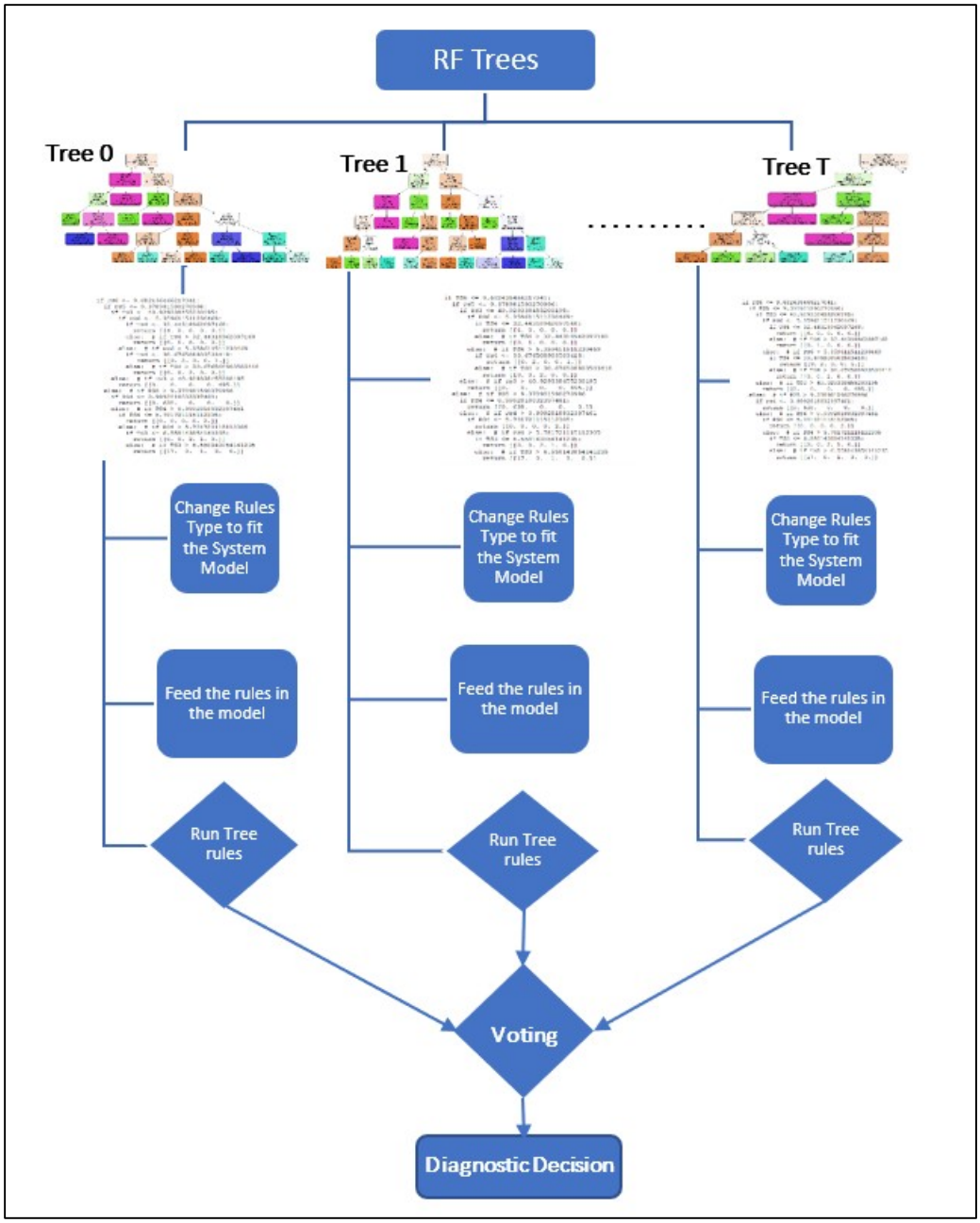


Figure 5. A Hybrid RF Approach between Data Driven and Model-based approaches

6. Discussion

In this work, the architecture of a hybrid FDD method between model-based and data-driven approaches to achieve FDD for component faults is introduced. In this hybrid method the data-driven part is represented by an optimized and hyperparameter tuned RF, in order to generate dynamic, diagnostic graphs that are later converted into a set of diagnostic rules and fed into a pre-defined system diagnostic model contributing as the model-based part of the proposed FDD system. The proposed approach provides a dynamic solution unlike other model-based FDD approaches. Additionally, the possibility to apply distributed computing on the diagnostic graphs and extracted rules which can reduce time and resources complexity comparing to traditional data-driven approaches. Moreover, the proposed method introduces a new methodology to approach RF in a model-based fashion, beyond its exclusive, ordinary application as a data-driven approach.

The data-driven part of this system is experimentally applied and analyzed using a multivariate time series sensor data collected from an actual hydraulic test rig. The applied data-driven part includes the RF creation, RF feature selection using non-zero feature importance, and RF pruning and hyperparameter tuning using 3-fold cross validation on a grid of variables selected using random search. Furthermore, the diagnostic rules in the form of nested if-else statements are practically extracted from RF as the diagnostic graph of this approach. The extracted rules can be converted into various forms and shapes depending on the nature of the system model subject of integration.

This work has successfully provided an extension and development for the model-based FDD approach introduced in [7], where the previous system is a domain-specific, and can only be applied to the model described in the ontology. Further, containing rigid knowledge preserved in the ontology as a set of semantic rules but later translated into a static, domain-specific, application-specific, with constrained reliability to the system's expert set of diagnostic rules. On the other hand, our proposed method provided dynamic and reliable diagnostic rules with combined advantaged of both data-driven and model-based approaches.

The proposed FDD system offers a vast majority of advantages and new insights. However, there is always room for improvements. Thus, some additional work and further modifications for the proposed system can be applied in the future. In this work the traditional RF algorithm is applied to serve as the dynamic rule generator in both the offline learning and the online update stage using the newly arrived sensor reading. Instead, considering online RF methods in the first place such as, Mondrian forests [38] or online incremental RF in [39] may reduce the training and update time. Moreover, in future work a full application of this approach will be introduced and examined in a practical research, where the extracted rules are converted into SPARQL queries to fit the ontology designed for the chosen system. Furthermore, a proper scheduler will be chosen to demonstrate the possibility of distributed computing using the extracted diagnostic rules at run-time.

Author Contributions: Conceptualization, A.M. and M.F.; methodology, A.M.; software, A.M.; validation, M.F.; formal analysis, A.M.; investigation, A.M.; resources, M.F.; data curation, A.M.; writing—original draft preparation, A.M.; writing—review and editing, M.F.; visualization, A.M.; supervision, M.F.; project administration, M.F.; funding acquisition, M.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the DFG research grants LO748/11-1 and OB384/5-1.

Conflicts of Interest: The authors declare no conflict of interest

References

1. Isermann, R.; Ballé, P. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Eng. Pract.* **1997**, *5*, 709–719.
2. Kanev, S.K. Robust fault-tolerant control.; 2004.
3. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 1–58.
4. Venkatasubramanian, V.; Rengaswamy, R.; Yin, K.; Kavuri, S.N. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Comput. Chem. Eng.* **2003**, *27*, 293–311.
5. Venkatasubramanian, V.; Rengaswamy, R.; Kavuri, S.N. A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. *Comput. Chem. Eng.* **2003**, *27*, 313–326.
6. Skliros, C.; Miguez, M.E.; Fakhre, A.; Jennions, I. A review of model based and data driven methods targeting hardware systems diagnostics. *Diagnostyka* **2018**, *20*, 3–21.

7. Mallak, A.; Behravan, A.; Weber, C.; Fathi, M.; Obermaisser, R. A Graph-Based Sensor Fault Detection and Diagnosis for Demand-Controlled Ventilation Systems Extracted from a Semantic Ontology. In Proceedings of the 2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES); 2018; pp. 000377–000382.
8. Definition of ONTOLOGY Available online: https://www.merriam-webster.com/dictionary/ontology (accessed on Aug 11, 2019).
9. Noshad, Z.; Javaid, N.; Saba, T.; Wadud, Z.; Saleem, M.Q.; Alzahrani, M.E.; Sheta, O.E. Fault Detection in Wireless Sensor Networks through the Random Forest Classifier. *Sensors* **2019**, *19*, 1568.
10. Zidi, S.; Moulahi, T.; Alaya, B. Fault Detection in Wireless Sensor Networks Through SVM Classifier. *IEEE Sens. J.* **2018**.
11. Lee, S.; Park, W.; Jung, S. Fault Detection of Aircraft System with Random Forest Algorithm and Similarity Measure Available online: https://www.hindawi.com/journals/tswj/2014/727359/ (accessed on Aug 26, 2019).
12. Wang, Z.; Zhang, Q.; Xiong, J.; Xiao, M.; Sun, G.; He, J. Fault Diagnosis of a Rolling Bearing Using Wavelet Packet Denoising and Random Forests. *IEEE Sens. J.* **2017**, *17*, 5581–5588.
13. Yang, B.-S.; Di, X.; Han, T. Random forests classifier for machine fault diagnosis. *J. Mech. Sci. Technol.* **2008**, *22*, 1716–1725.
14. Ccopa Rivera, E.; Swerdlow, J.J.; Summerscales, R.L.; Uppala, P.P.T.; Maciel Filho, R.; Neto, M.R.C.; Kwon, H.J. Data-Driven Modeling of Smartphone-Based Electrochemiluminescence Sensor Data Using Artificial Intelligence. *Sensors* **2020**, *20*, 625.
15. Gajowniczek, K.; Grzegorczyk, I.; Ząbkowski, T. Reducing False Arrhythmia Alarms Using Different Methods of Probability and Class Assignment in Random Forest Learning Methods. *Sensors* **2019**, *19*, 1588.
16. Rokach, L.; Maimon, O. *Data mining with decision trees: theory and applications*; Second edition.; World Scientific: Hackensack, New Jersey, 2015; ISBN 978-981-4590-07-5.
17. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. Classification and Regression Trees.; 1983.
18. Shalev-Shwartz, S.; Ben-David, S. Understanding Machine Learning: From Theory to Algorithms.; 2014.
19. Rokach, L.; Maimon, O. Top-down induction of decision trees classifiers - a survey. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2005**, *35*, 476–487.
20. Anderson, B. *Pattern Recognition: An introduction*; Scientific e-Resources, 2019; ISBN 978-1-83947-239-8.
21. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*; Springer Series in Statistics; 2nd ed.; Springer-Verlag: New York, 2009; ISBN 978-0-387-84857-0.
22. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140.
23. Tin Kam Ho Random decision forests. In Proceedings of the Proceedings of 3rd International Conference on Document Analysis and Recognition; 1995; Vol. 1, pp. 278–282 vol.1.
24. RANDOM FORESTS Trademark of MINITAB, LLC - Registration Number 3185828 - Serial Number 78642027 :: Justia Trademarks Available online: http://trademarks.justia.com/786/42/random-78642027.html (accessed on Jan 18, 2020).
25. Tin Kam Ho The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 832–844.

26. Bryll, R.; Gutierrez-Osuna, R.; Quek, F. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognit.* **2003**, *36*, 1291–1302.
27. UCI Machine Learning Repository: Citation Policy Available online: https://archive.ics.uci.edu/ml/citation_policy.html (accessed on Feb 4, 2020).
28. Helwig, N.; Pignanelli, E.; Schütze, A. D8.1 - Detecting and Compensating Sensor Faults in a Hydraulic Condition Monitoring System. *Proc. Sens. 2015* **2015**, 641–646.
29. Helwig, N.; Pignanelli, E.; Schütze, A. Condition monitoring of a complex hydraulic system using multivariate statistics. In Proceedings of the 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings; 2015; pp. 210–215.
30. Schneider, T.; Helwig, N.; Schütze, A. Automatic feature extraction and selection for classification of cyclical time series data. *Tm - Tech. Mess.* **2017**, *84*, 198–206.
31. Louppe, G.; Wehenkel, L.; Sutera, A.; Geurts, P. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems 26*; Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q., Eds.; Curran Associates, Inc., 2013; pp. 431–439.
32. Beware Default Random Forest Importances Available online: http://explained.ai/decision-tree-viz/index.html (accessed on Feb 8, 2020).
33. Probst, P.; Wright, M.; Boulesteix, A.-L. Hyperparameters and Tuning Strategies for Random Forest. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2019**, *9*.
34. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
35. Zabinsky, Z.B. Random Search Algorithms. In *Wiley Encyclopedia of Operations Research and Management Science*; American Cancer Society, 2011 ISBN 978-0-470-40053-1.
36. sklearn.model_selection.RandomizedSearchCV — scikit-learn 0.22.1 documentation Available online: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html (accessed on Feb 9, 2020).
37. SPARQL. *Wikipedia* 2020.
38. Lakshminarayanan, B.; Roy, D.M.; Teh, Y.W. Mondrian Forests: Efficient Online Random Forests. *ArXiv14062673 Cs Stat* **2015**.
39. Hassab Elgawi, O.; Hasegawa, O. Online incremental random forests. In Proceedings of the 2007 International Conference on Machine Vision; 2007; pp. 102–106.