






Article

SOPHIA: An Event-Based IoT and Machine Learning Architecture for Predictive Maintenance in Industry 4.0

Matteo Calabrese ^{1,†}, Martin Cimmino ^{1,†}, Francesca Fiume ^{1,†} , Martina Manfrin ^{1,†},
Luca Romeo ^{2,*,†} , Silvia Ceccacci ^{3,†} , Marina Paolanti ^{2,†} , Giuseppe Toscano ^{4,†},
Giovanni Ciandrini ^{4,†}, Alberto Carrotta ^{4,†}, Maura Mengoni ^{3,†}, Emanuele Frontoni ^{2,†}
and Dimos Kapetis ^{1,†} 

¹ Accenture Digital, ICEG Artificial Intelligence Center of Excellence (CoE), viale Monza 265-259, 20126 Milan, Italy; m.calabrese@accenture.com (M.C.); martin.cimmino@accenture.com (M.C.); francesca.fiume@accenture.com (F.F.); martina.manfrin@accenture.com (M.M.); dimos.kapetis@accenture.com (D.K.)

² Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, viale Breccie Bianche 12, 60131 Ancona, Italy; m.paolanti@pm.univpm.it (M.P.); e.frontoni@staff.univpm.it (E.F.)

³ Dipartimento di Ingegneria di Ingegneria Industriale e Scienze Matematiche, Università Politecnica delle Marche, viale Breccie Bianche 12, 60131 Ancona, Italy; s.ceccacci@pm.univpm.it (S.C.); m.mengoni@univpm.it (M.M.)

⁴ BIESE GROUP SpA, viale della Meccanica 16, 61122 Pesaro, Italy; giuseppe.toscano@biesse.com (G.T.); giovanni.ciandrini@biesse.com (G.C.); alberto.carrotta@biesse.com (A.C.)

* Correspondence: l.romeo@univpm.it; Tel.: +39-0712-204-900

† These authors contributed equally to this work.

Received: 23 March 2020; Accepted: 7 April 2020; Published: 9 April 2020



Abstract: Predictive Maintenance (PdM) is a prominent strategy comprising all the operational techniques and actions required to ensure machine availability and to prevent a machine-down failure. One of the main challenges of PdM is to design and develop an embedded smart system to monitor and predict the health status of the machine. In this work, we use a data-driven approach based on machine learning applied to woodworking industrial machines for a major woodworking Italian corporation. Predicted failures probabilities are calculated through tree-based classification models (Gradient Boosting, Random Forest and Extreme Gradient Boosting) and calculated as the temporal evolution of event data. This is achieved by applying temporal feature engineering techniques and training an ensemble of classification algorithms to predict Remaining Useful Lifetime (RUL) of woodworking machines. The effectiveness of the proposed method is showed by testing an independent sample of additional woodworking machines without presenting machine down. The Gradient Boosting model achieved accuracy, recall, and precision of 98.9%, 99.6%, and 99.1%. Our predictive maintenance approach deployed on a Big Data framework allows screening simultaneously multiple connected machines by learning from terabytes of log data. The target prediction provides salient information which can be adopted within the maintenance management practice.

Keywords: predictive maintenance; machine learning; Remaining Useful Lifetime; feature engineering; Big Data platform

1. Introduction

The increasing availability of Big Data technology platforms and data-driven applications are changing the way decisions are taken in the industry in important areas such as scheduling [1], maintenance management [2,3], and quality improvement [4–6]. In manufacturing industry machines and systems become more advanced and complicated. End-users demand comprehensive maintenance

service in their production equipment to ensure high availability preventing machine downtimes. In this context, machine learning can be efficiently used for optimal maintenance decision-making. Most of the companies and manufacturers possess huge amounts of sensor, process, and environment data. Combining the data with information about the failures creates useful train data sets for predictive maintenance.

Approaches to maintenance management are grouped into three main categories which, in order of increasing complexity and efficiency [7], are: (i) Run-to-Failure (R2F), (ii) preventive maintenance and (iii) predictive maintenance. R2F is the simplest approach dealing with maintenance interventions which are performed only after the occurrence of failures. Preventive maintenance (PM) comprises all the operational techniques and actions required to ensure machine availability and to prevent a “machine-down” failure. PM is defined as regularly scheduled maintenance actions based on average failure rates driven by time-, meter-, or event-based triggers. A properly implemented PM strategy can provide many benefits to an organization by extending equipment life, optimizing resource expenditures, and balancing work schedules. However, poor maintenance strategies can reduce a plant’s overall machine productive capacity by 5% to 20% [8]. Extensive PM programs require many labor resources, and there is often a probability of performing excessive maintenance that has no positive impact on the equipment [9]. Therefore, it is difficult to determine the optimal level of PM, and it may require years of maintenance actions and data collection before payback is realized [10]. Such maintenance is often carried out separately for every component, based on its usage or on some fixed schedule. Unlike traditional maintenance methods, Predictive Maintenance (PdM), could play a central role for asset utilization, service, and after-sales in the realizing Industry 4.0 new technological services. PdM is defined as a process of determining maintenance actions according to regular inspections of an equipment asset’s physical parameters, degradation mechanisms, and stressors to correct problems before machine-down occurs [9]. Standard EN 13306:2001 [CEN: “Maintenance terminology”, European Standard EN13306, 2001] defines PdM as “condition-based maintenance carried out following a forecast derived from the analysis and evaluation of significant parameters of the degradation of the item”. Contrary to the classic PM programs, PdM could improve the performance of the equipment, strengthening the business model of companies. Thanks to including a set of sensing, condition-based monitoring (CBM), predictive analytics and distributed systems technologies, it is possible to provide remote technical help based on continuous monitoring. PdM works particularly well for systems easy to monitor and have easily identifiable characteristics that can be statistically analyzed to determine the Remaining Useful Lifetime (RUL) [11]. Notwithstanding research unanimously states that implementing of PdM can bring enormous benefits to businesses in terms to reduce life-cycle costs [12] and increasing product reliability [13], advanced maintenance technologies have not yet been well implemented in the manufacturing industry. This depends on various reasons, including the difficulty of gaining appropriately condition monitoring (CM) data (e.g., vibrations, currents, temperatures, etc.) and collect run to failure data in the industrial production environment [14]. In fact, this requires the machines equipped with sensors and proper data acquisition systems and often implies huge investments to redesign sub-systems of the machines. However, many types of equipment, including CNC machine tools, can already provide in real time massive state data about machining process and a large amount of event data related to errors and faults generated by the diagnostic systems embedded by their controllers. All these data together represent a huge wealth of information that nowadays machine tool manufacturers can easily access thanks to the new Internet of Things (IoT) and Information and Communication Technologies (ICT) introduced with the Industry 4.0.

Based on our knowledge, current research neglects the importance of such data for PdM in manufacturing industry; very few studies explored the possibility to perform equipment failure prediction from event log data in the case of electronic equipment, such as medical devices [15] and automated teller machines [16]. No studies proposed similar approaches to predict the failure of specific mechanical components of machine tools in the context of the manufacturing industry. Perhaps,

non-observance of event data may result from the erroneous belief that event data is not valuable as long as the CM indicators seem to work well in reducing equipment failures [17]. However, taking advantage of this data source would allow alternative low-cost PdM, without requiring additional sensors to be implemented on the machines. The idea of using equipment logs to predict faults poses several issues that have not yet been fully explored. In particular, determining predictive features poses a major challenge, as the logs contain a massive amount of data that rarely includes explicit information for failure prediction [15]. The management of these volumes of data requires a system capable of incorporating the entire technology stack: the extraction-transformation-load, the data filtering, and advanced machine learning algorithms to identify hidden patterns of operational, usage and maintenance data log files to predict machine downtime. To achieve this, Big Data frameworks come in useful for analyzing the data more efficiently and deploy PdM in order to be executed in a scalable cloud computing environment.

In this context, the main contribution of this research is to present a methodology for PdM, taking advantage of Big Data information already provided by machine tool data log systems, without the need to install sensors into the machines to collect specific CM data. This work presents a PdM approach to adopt a data-driven solution deployed in a scalable cloud computing environment. We describe a machine learning application for PdM applied to the ball-bearing component of the Electros spindle (ES) machine. We present a data-driven approach based on multi-step machine learning pipeline comprising (i) log file parser development (ii) feature engineering (iii) model building and model evaluation (iv) model deployment and monitoring. By the application of this computation pipeline, we tested the hypothesis that the aggregated event-driven data (errors and warning events) are associated with machine-down and can be qualified as predictive markers or KPIs for machine down of woodworking machines.

2. Related Work

As evidenced by many research reviews [14,17–19], during the past twenty years, enormous efforts have been made to improve CBM systems with increasingly effective diagnostic and prognostic capabilities. The approaches predominantly used for the development of PdM systems can be classified in knowledge-driven and data-driven [14,19].

Knowledge-driven approaches typically make use of “a priori” human expertise or involve building models based on comprehensive knowledge about system physics. When fault models of the system and their progression are available, model-based approaches provide the most accurate and effective results [19]. However, as the system complexity and uncertainties increase, it can be very difficult and costly to adopt such approaches. Data-driven approaches represent the most suitable solution in all practical cases where it is easier to gather data than building accurate expert systems or physics models [18]. They attempt to derive models directly from historical records, exploiting machine learning algorithms: the basic idea is to learn system behavior by observing variations in nominal operating conditions showing faulty operating states and to extrapolate knowledge to determine the RUL of certain components. The most commonly used learning approaches include Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Bayesian networks (BNs), and Hidden Markov Models (HMMs) [14,19,20]. ANN is the most commonly used Artificial Intelligence technique in machinery RUL prediction [14], mainly because of its strong ability for nonlinear simulation, strong robustness and self-study ability [20]. However, this method often requires large numbers of high-quality training data and it is difficult to be generalized among different contexts, as its structure and parameters are generally initialized randomly or specified manually [14]. BN is a probabilistic acyclic graphical model [21], where each node represents a random variable that can be continuous or discrete. The expert knowledge about the domain is needed to identify those random variables and build the graph [22]. BN is a multiple state model with the ability to test several outputs in the same model, but it can only be used to predict failure times for previously anticipated faults, with anticipated symptoms. Dynamic Bayesian Network (DBN) extends BN. It is a general state-space

model to describe stochastic dynamic systems [23]. DBN is used for prediction failures in industrial plants [24]. While DBN is a potential tool to predict failures in advance, its models are applicable so far for a simple network with a few variables. Several works have also been developed using HMM to estimate the sequences of hidden degradation states of a system before a failure occurs [25–27]. HMM model accuracy outcome depends on the quality of temporal data available and are not suitable for a large number of variables. To collect information about the operating conditions of critical components of mechanical equipment, proper sensors are needed for recording and monitoring CM data (e.g., vibration, temperature, current, etc.). Some studies refer both on CM data and event data. For example, Canizo et al. [28] presented a Big Data analytics approach for the renewable energy field. In particular, they developed a PdM application for wind turbines by using a Big Data processing framework to generate data-driven predictive models that are based upon historical operational data (e.g., power, wind speed, rotor speed, and generator speed recorded) and system status data, previously stored in the cloud.

Taking into account the increasing amount of data, there is growing interest in the application of machine learning and reasoning for predictive maintenance. In particular, machine learning pros may lie the foundation for overcoming several challenges including the need to integrate data from various sources (sources heterogeneity) and systems, the need to provide accurate prediction model [29] and the need for real-time monitoring demand dealing with latency, and scalability [30]. Hence, the major pros of the ML model in this context can be justified in terms of predictive performance, computation effort and interpretability. However, the use of machine learning in this context may raise other challenges like obtaining training data; dealing with the dynamic operating conditions (domain shifts) and the need to select the more appropriate method for each industrial case. Only in recent years, some data-driven approaches for PdM based only on event log data have been proposed for several equipments. For example, Sipos et al. [15] report the application for multiple instance learning (MIL) to build predictive models from log data for medical equipment. MIL is a variation of supervised learning where a single class label is assigned to a bag of instances (bags are labeled positive or negative). Given bags got from different equipment and at different dates, the goal is to build a classifier that will label either unseen bags or unseen instances correctly. However, based on the best of our knowledge, no studies proposed similar approaches to achieve PdM to predict the failure of specific mechanical components (e.g., roller bearings, motor inverters, etc.), taking advantage of CNC machine tool data log systems.

In our study, we adapted our PdM application into a Big Data environment using cloud computing and Big Data frameworks to provide the method with the ability to scale and to process the data of hundreds of thousands of connected machines. We build a computational pipeline to deal with high dimensional data and to model complex interaction event-based error aggregates parsed from unstructured log files. To achieve this, we tested the hypothesis that aggregated features computed by feature engineering on temporal data could be efficiently modeled with machine learning classifiers. The proposed application can help to enable PdM and root cause analysis of machine down of woodworking machines.

3. Materials and Methods

3.1. Log File Data Collection

We concentrated our efforts on the broken ball bearing component which causes the machine down of woodworker Rover devices. Log files were collected having at least five months of historical data and obtained the standard output (stdout) log files. The collected stdout log files data set was from 14 Rover machines: 5 ES with the broken ball bearing component, and 9 ES without ball bearing (control group) were subjected for data processing as described in the following paragraphs.

3.2. Big Data Architecture

The Big Data architecture is reported in Figure 1. Big Data design of the PdM application comprised three major modules: (i) data acquisition (ii) data processing and data science module and (iii) predictive monitoring. For data acquisition, Azure Blob Storage (<https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-hadoop-use-blob-storage>) was used. Azure storage is a distributed, robust, general-purpose storage solution that integrates with Azure HDInsight. HDInsight (Azure Hadoop Distribution) uses a blob container in Azure Blob Storage as the default file system for the cluster. Through a Hadoop distributed file system (HDFS) interface, the full set of components in HDInsight can operate directly log files stored as blobs. Data are retrieved through Azure Blob Storage REST APIs service using the HDFS interface.

Apache Spark (<https://spark.apache.org/>) (in Python version called pySpark) was used for data processing (Figure 1B). Apache Spark is a fast, general purpose memory engine for large-scale data processing and feature engineering. These features make Apache Spark well-suited to process and analyze large volumes of log files. Apache Spark is also a general-purpose and very flexible platform. Thanks to this last feature it was possible to install the pySpark libraries to run the machine learning algorithm. The PdM application uses two data processing types: offline processing to generate predictive models based on historical log data files and online processing. The Python language is used for the offline mode, while pySpark is employed for the production environment (Figure 1B). Online processing of the entire pipeline Spark job was scheduled to run the entire pipeline through the Linux Cron scheduler (every 24 h) and to update with new data the prediction probabilities of the machine down. Finally, the updated prediction probabilities for machine down failure were daily visualized and monitored through a front-end web application.

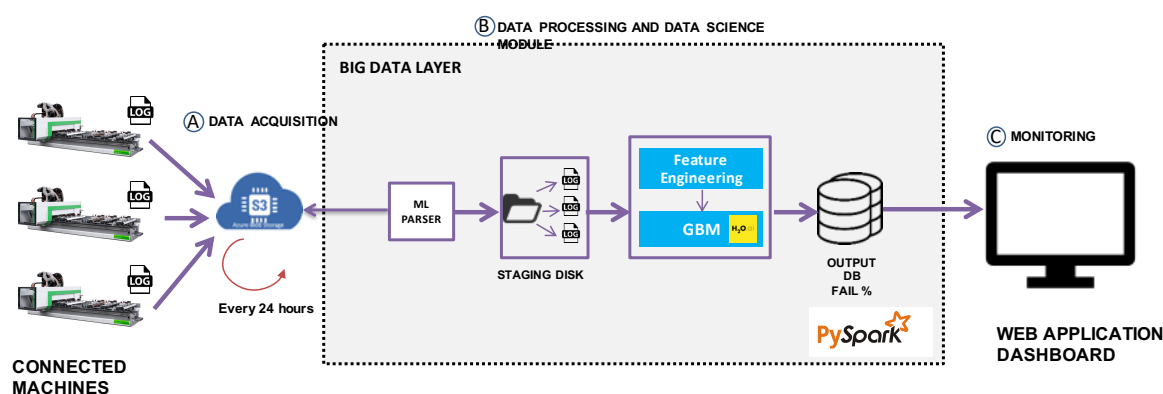


Figure 1. High-level pipeline architecture. The pipeline architecture illustrates the architecture that characterizes the Predictive Maintenance (PdM) application. (A) Data acquisition step is performed through Azure Blob storage service. (B) Apache Spark is used for data preprocessing, parsing of log files and predictive modeling. (C) The predicted probabilities of machine down are visualized by a dedicated dashboard to evidence the trend of connected machines.

3.3. Data Science Module

The data science module is summarized in Figure 2B. Our methodology can be encapsulated in a protocol that has two main components: Log file parsing and Machine Learning (ML) module. The main steps of the current protocol are: (i) log-file parsing, (ii) feature engineering for selected group error groups (Table A2), (iii) model building, (iv) model evaluation.

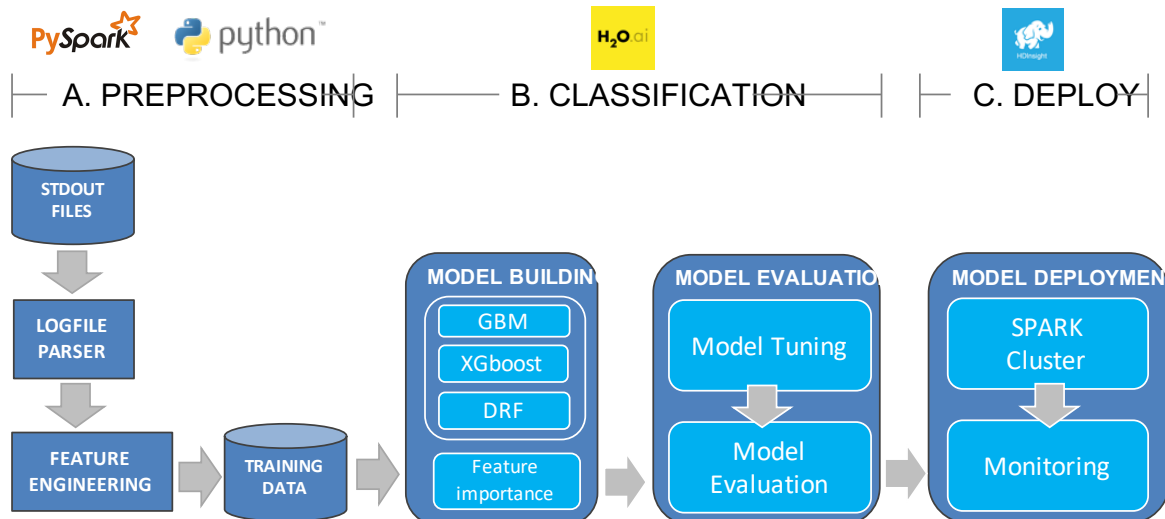


Figure 2. Data processing and data science module overview Flowchart diagram indicates the main steps of the data mining pipeline analysis. (A) During the pre-processing step, input stdout files were preprocessed within the developed parser module in order to create the training data set. (B) In the classification step, tree-based classifiers (Gradient Boosting Machine (GBM), Extreme Gradient Boosting (XGBoost), and Distributed Random Forest (DRF)) were applied to build and evaluate their performance by H2O.ai framework. (C) The optimal classifier was deployed within a distributed environment of Hadoop Spark cluster. The performance metrics of the optimal classifier were monitored to validate the model metrics.

3.3.1. Log-File Parsing Module

Event triggers and log file timestamp were parsed using ad hoc regular expressions (Figure 3A). In total, 104 selected group errors and warnings were selected and grouped into five major error groups: emergency, security control, unexpected stop (KO) error, inverter, overheating, tool change (Table A2). The target variable was expressed as RUL and estimated by calculating the remaining number of days to the failure event based as provided by the internal service records (Figure 3B,C). The RUL variable was discretized into a binary variable and tested for several windows period before machine down ($W = \{30,20,10\}$ days). The optimal binary RUL split was evaluated and tested into the classification step.

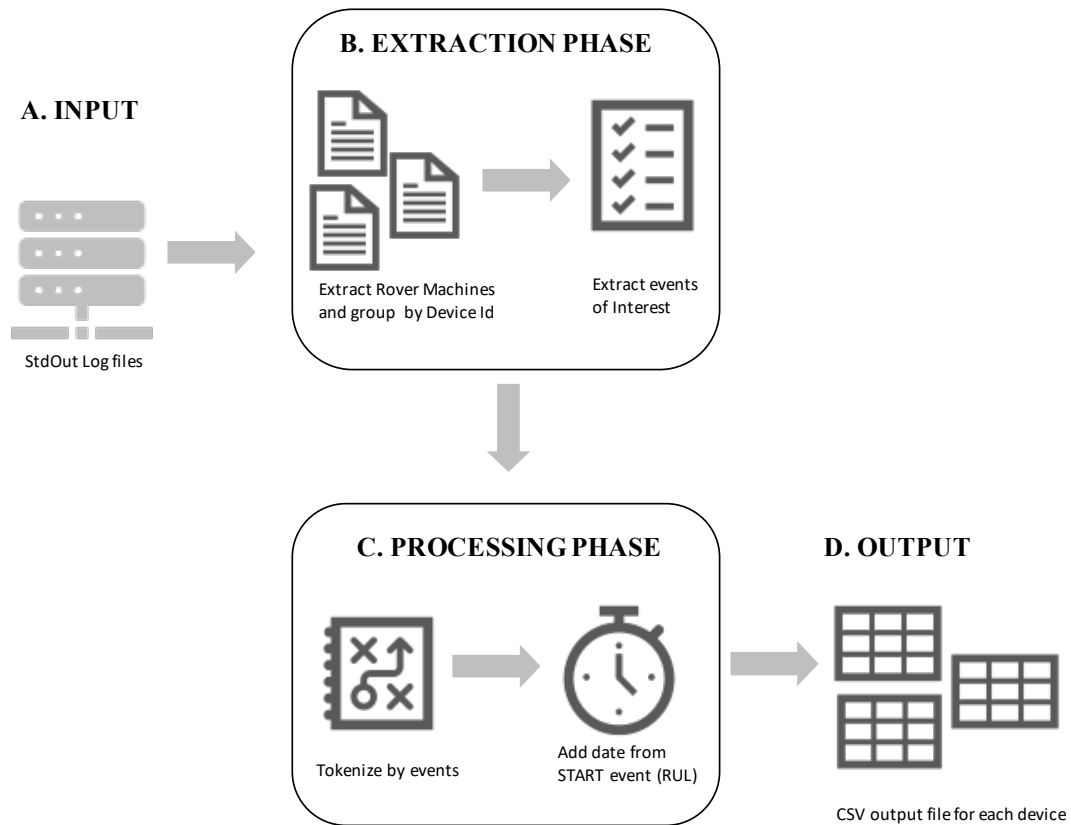


Figure 3. Parser Flowchart development Flowchart diagram indicates the phases of the parser development (A,B) During the extraction phase, unstructured stdout log file we collected for each device and parsed to extract events of interest and timestamp. (C,D) In the processing phase, the extracted event triggers and timestamp were tokenized and formatted into .csv output for each Rover device.

3.3.2. Feature Engineering

We used a ‘Bag-of-words’ approach using Scikit-learn’s CountVectorizer [31] to count frequency of events (Figure 2A). Feature engineering was applied, to construct the final training data set. In this step, for each classification group error (Table A2), the rolling mean of size ‘W’ ($W = \{30,20,15,10\}$ days) was applied to calculate the lagged features (Table 1).

Table 1. Lagged feature variables obtained with feature engineering. KO: unexpected stop error, SOS: errors that contain the word “Emergency”.

Lag Features ($W = \{30,20,15,10\}$ days)	Description
Programs	Reports the average frequency of program aggregates of size W
Security control errors	Reports the average frequency of error aggregates of size W
Inverter errors	Reports the average frequency of inverter error aggregates of size W
KO error	Reports the average frequency of KO error aggregates of size W
SOS error	Reports the average frequency of SOS aggregates of size W
Emergency errors	Reports the average frequency of emergency errors aggregates of size W
Overheating error	Reports the average frequency of overheating error aggregates of size W
Tool change	Reports the average frequency of tool change aggregates of size W
Global errors	Report the average frequency all error (security control, inverter, KO, SOS, overheating, tool change) aggregates of size W
SOS program error	Report the average frequency number of programs within of size W

3.3.3. Model Optimization

On the training set, cross-validation (CV) was used to estimate the optimal values of hyperparameters. The optimal parameter values were determined using a grid-search on the training set. The grid search was performed over the several ranges of the following parameters: max depth, learning rate, number of trees, column sample rate and sample rate as reported in the Table A3. For each approach, the optimized set of hyperparameters was then used to train the classifier using the training group; the performance of the resulting classifier was then evaluated on the testing set. In this way, we achieved unbiased estimates of the performances of each method.

3.3.4. Machine Learning Model Building

Using the H2O.ai python package (<https://www.h2o.ai>) we trained and tested nine machine learning models with different binary RUL targets ($W = \{30,20,10\}$ days) comprising Extreme Gradient Boosting (XGBoost), Distributed Random forest (DRF) and Gradient Boosting Machine (GBM). The key advantage of tree-based models is that they are easy to interpret, since the predictions are given by a set of rules [32,33].

We performed experiments with other state-of-the-art machine learning methodologies (i.e., SVM with linear and Gaussian Kernel, nearest-neighbor [NN] classifier and decision tree [DT]). In our task, Extreme Gradient Boosting (XGBoost), Distributed Random forest (DRF) and Gradient Boosting Machine (GBM) achieved higher performance (in terms of accuracy, precision, and recall) than SVM and DT. We have not considered the neural network-based model in our comparisons, because the potential of neural network models may be limited by the interpretability of the model, which does not always allow to perform the pattern localization [34]. On the other hand, the key advantage of tree-based models is that they are easy to interpret since the predictions are given by a set of rules. The knowledge of these rules and how the prediction is achieved are key information that may support the operator for the diagnosis and prognosis procedure in the context of PdM.

3.4. Experimental Procedures and Metrics

To obtain unbiased estimates of the performances, the data set divided into two groups: a training set and testing set with a 70-30 split. The training set (70%) was used to determine the optimal values of the hyperparameters of each method and to train the classifier. The testing set was then only used to test classification performances. On the training set, CV was used to estimate the optimal values of hyperparameters. The optimal parameter values were determined using a grid-search on the training set. The grid-search was performed over the several ranges of the following parameters: max depth, learning rate, number of trees, column sample rate and sample rate as reported in the Table A3. For each approach, the optimized set of hyperparameters was then used to train the classifier using the training group; the performance of the resulting classifier was then evaluated on the testing set. The ML model, which achieved the highest expected value of accuracy (see (1)), was selected as the optimal classifier tested for several periods before machine down ($W = \{30,20,10\}$ days).

The performance of the ML model was evaluated in terms of predictive accuracy and model interpretability. Predictive accuracy of the proposed approach was performed according to the following measures:

- Accuracy, Recall and Precision:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (1)$$

$$Recall = TP / (TP + FN) \quad (2)$$

$$Precision = TP / (TP + FP) \quad (3)$$

where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

- Receiver operating characteristic (ROC): is designed by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. It illustrates the performance of a binary classifier as its threshold is changed. We used an area under the ROC curve (AUC) to compare the performance of classifiers.
- Confusion matrix: the square matrix that shows the type of error in a supervised task. For the considered binary task we show *TP*, *TN*, *FP* and *FN* values.

In terms of model interpretability, the variable importance was used to determine the relative influence of each variable on the classification task for the three models GBM, RF, and XGBM.

4. Experimental Results

The analysis was based on a computational pipeline protocol (Figure 2), comprising the following step: (i) pre-processing step (Log file parser, Feature engineering) and (ii) classification step (Model Building and Model evaluation) and (iii) model deployment (Hadoop Spark Cluster and monitoring).

4.1. Log File Parser

In the pre-processing step, we developed a log file parser to extract events and pre-processing the data (Figure 2A). Stout log files are a collection of events recorded by various applications running on the ES equipment. The input log file consists of a timestamp expressed in milliseconds indicating when the events occur, message texts (either fully unstructured or generated from a template) describing the event, represented with an alphanumeric code to indicate different event category or groups event variations which reflect the developers' original idea about what are the valuable states to report. Events can be aggregated into functional units called program units as defined between specific start and stop flags (Table A1). During the extraction phase (Figure 3B), we applied a set of regular expressions to (i) extract program units and (ii) extract specific event codes, including event codes (PLC), error codes, inverter codes, etc. (Table A2). In the processing phase, (Figure 3C) each record was aggregated into program units and calculated frequency of all event codes. The target variable (Figure 3C) was expressed as RUL and was estimated by calculating the remaining number of days to the failure event based as provided by the internal service records.

4.2. Feature Engineering

We applied a feature engineering technique to create more informative variables to our training data set using sliding windows method to calculate novel variables called lag features. Sliding windows are fixed size windows that are used for aggregate computations. For each data point in a temporal sequence, aggregates are computed from the data points in a pre-defined window. In our training data set, for each record, a sliding window of size 'W' was chosen as the number of units of time to compute the lag features for daily program units (Table 1). Lag features of the program were computed using the W periods before the event of machine down failure were calculated within four windows: ($W = \{30,20,15,10\}$ days). The final data set (Figure 3D), consisting of 14 cases (five with machine failure and nine without machine failure), returned 44 aggregated features and 2975 records which are submitted to the classification step (Figure 2B).

4.3. Classification Results

In the classification step, we trained, tuned, and evaluated three tree-based models: Distributed Random Forest (DRF), Gradient Boosting Machine (GBM) and Extreme Gradient Boosting (XGBoost). For each model the optimal values of the hyperparameters were identified using a grid-search method (Table A3). Classification accuracy was estimated by means of four-fold CV in training data set (70%) and validation on an independent testing data set (30%). Since the GBM model achieved for the 30-day RUL target the highest expected value of accuracy in the testing data set, it was selected as the optimal classifier. The classification accuracy values of all three models were listed in Table 2. The accuracy,

sensitivity, and specificity were 98.9%, 98.6% and 98.3% in the testing group, respectively (Table 2, Figure 4A,B). The Area Under Curve of the ROC (AUC) was 0.92 in the testing data set (Figure 4B). XGboost and DRF models achieved lower metric values in the testing set (Table 2). The confusion matrix for this GBM classifier supported these observations as well (Figure 4A). This classifier demonstrated a reliable performance showing that can predict the machine down caused by the ball bearing component with high accuracy.

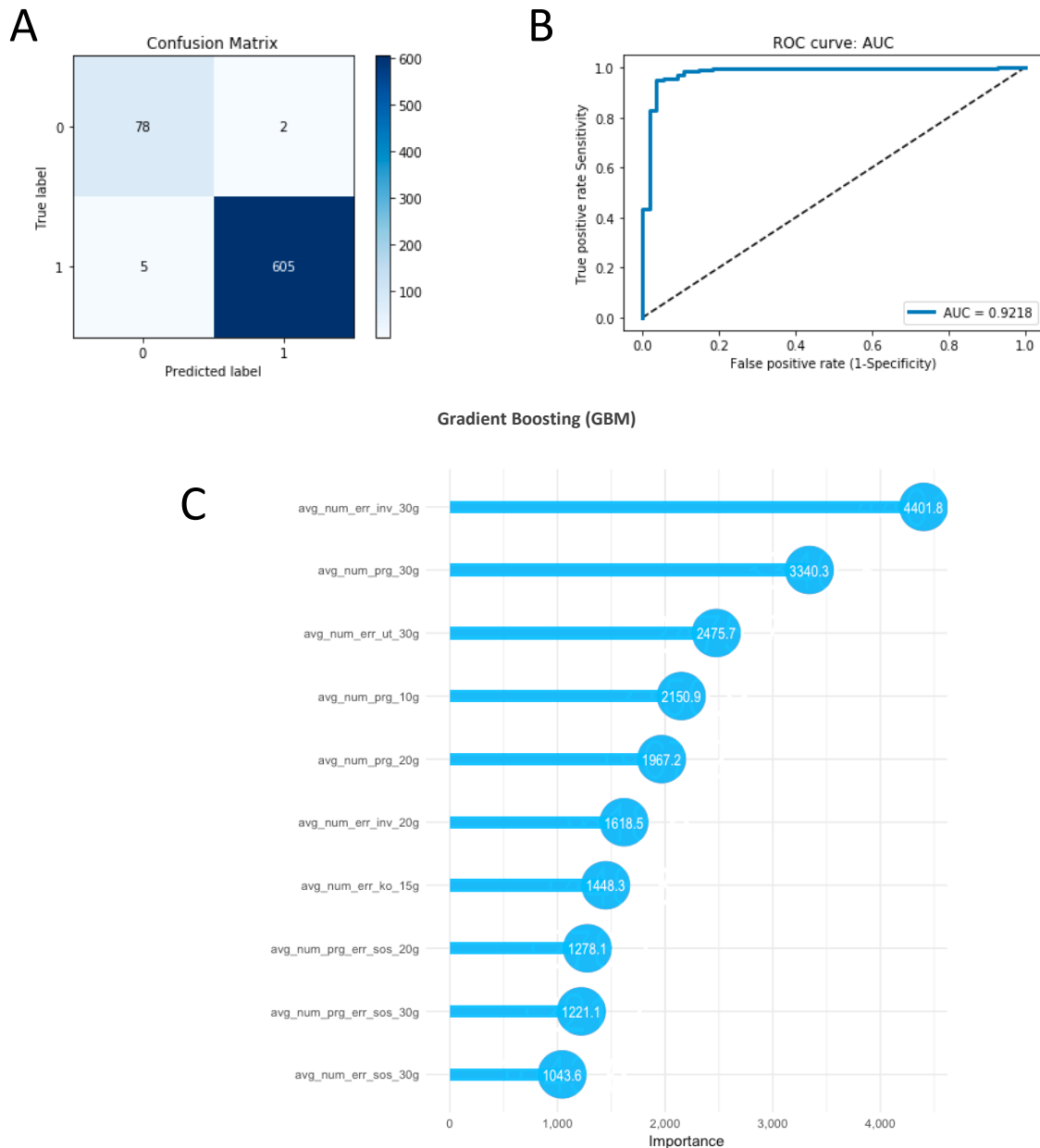


Figure 4. Classification metrics of Gradient Boosting Machine (GBM) classifier (A) Confusion matrix of GBM Classifier on testing data set. GBM classified 78 out of 81 records considering the machine down period (30-day Remaining Useful Lifetime (RUL) machine-down) and 605 out of 610 without machine down. (B) Receiver operating curve (ROC) of the machine down within 30-day RUL period and without machine down (control group; > 30-day RUL). The area under the curve was 92.18%. (C) Variable importance of GBM classifier. The horizontal plot shows top 10 variables selected from the model to classify machine down against the control group.

Table 2. Accuracy, Recall and Precision metrics of GBM, DRF and XGboost calculated on training cross-validation (CV) and testing data sets. All metric values (accuracy, recall, and precision) are multiplied by 100. The Standard deviation of training CV is reported in parentheses. The bold letter shows the optimal classifier of each model (GBM, DRF and XGboost) tested on different RUL.

Model	RUL	Accuracy		Recall		Precision	
		Training	Testing	Training	Testing	Training	Testing
Distributed Random Forest (DRF)	30	97.4 (0.4)	96.8	98.8 (0.4)	96.5	98.3 (0.3)	100
	20	97.8 (0.2)	96.5	99.3 (0.2)	96.5	98.4 (0.4)	99.3
	10	98.2 (0.5)	96.5	99.6 (0.2)	96.9	98.5 (0.4)	99.3
Gradient Boosting Machine (GBM)							
Gradient Boosting Machine (GBM)	30	98.2 (0.2)	98.9	99.6 (0.1)	99.6	98.3 (0.3)	99.1
	20	98.7 (0.4)	97.8	99.5 (0.2)	98.7	99.1 (0.2)	98.8
	10	98.5 (0.8)	97.8	93.3 (0.2)	97.9	99.2 (0.6)	99.8
Extreme Gradient Boosting (XGBoost)							
Extreme Gradient Boosting (XGBoost)	30	98.1 (0.5)	98.8	99.5 (0.1)	100	98.4 (0.7)	98.7
	20	98.3 (0.7)	96.3	99.7 (0.01)	100	98.4(0.7)	96.2
	10	98.4 (0.6)	98.8	99.6 (0.1)	99.8	98.8 (0.7)	99.8

4.4. Model Feature Importance

The high interpretability of the GBM model allows extracting the importance of each feature to localize the most discriminative predictors. The H2O GBM algorithm [35] calculates the importance of each feature based on whether that variable was selected during splitting in the tree building process and how much the squared error improved as a result. Figure 4C shows a horizontal bar-plot, with only the top 10 variables which indicate which variable the GBM has learned are important in distinguishing machine down failure against the control group. The features are sorted according to the decreasing order to select the most relevant ones, assuming that the feature with the highest score is the most discriminant one. The most significant variables provided by the optimal model (GBM) we observe the top three variable the GBM belong to the invert error group (avg_num_err_inv_30), the number of daily programs (avg_num_err_30) and the tool change (avg_num_err_ut_30) whereas DFR and XGBM belong to average number of programs (avg_num_pgr_30, avg_num_pgr_15 and avg_num_pgr_10) and invert error (avg_num_err_inv_30) (Figure 4C and Figure A1 in Appendix A).

5. Platform Integration: PdM Application and Machine Down Monitoring

In the deployment phase (Figure 2C), the machine learning was deployed on Azure HDinsight Spark Cluster with Pyspark components. This architecture was composed of two master nodes and two slave nodes and other services like an SQL, and distributed storage. BIESSE manufacturing machines were connected over the internet network and providing monitoring of machine status. As shown in Figure 1, log files were retrieved from blob storage (Figure 1A) and pre-processed by the log file parser (Figure 1B). Raw log files were provided in a compressed format. Log files were then decompressed, parsed and saved in a staging layer and submitted to the Data Science module (Figure 1B) the machine down failure probabilities. To update prediction probabilities of machine down predictions an agent scheduler (every 24 h) was scheduled to run the Spark job pipeline. Prediction probabilities were visualized within the dashboard to monitor the status of machine down (Figure 1C).

6. Discussions

One of the challenges for the Industry 4.0 approach is to design and develop an embedded smart system to enable the health status of the machine. PdM has been featured as a key theme of Industry 4.0 which application allows reducing unscheduled downtime and consequently improve

productivity and reduce production cost. Our work was focused on the design and development of a computational pipeline for classifying the health status of the machine-based event information by taking into advantage log files event-based errors. Through the application of advanced analytics on new data streams either in the cloud-connected machine benefits from machine learning algorithms to perform PdM.

Our approach was developed with active involvement from domain experts and was evaluated and shown to be effective by both machine learning and domain standards. The proposed approach allowed screening simultaneously multiple connected machines, thus providing in day-by-day monitoring that can be adopted with maintenance management. This is achieved by applying temporal feature engineering techniques and training tree-based classification algorithms to predict the RUL of woodworking machines. The effectiveness of the proposed approach is demonstrated by testing an independent sample of additional woodworking machines. The GBM classifier correctly classified 78 out of 81 records considering the machine down period (30-day RUL machine-down) and 605 out of 610 without machine down, yielding 98.2% accuracy, 98.6% recall and 98.3% precision (Figure 4A). The area under the ROC curve value for the scores was 92.1% (Figure 4B).

We showed that starting from unstructured log files, trigger events can be efficiently exploited to enable a PdM system. This is due to several unique aspects of log files: containing a timestamp and can be viewed both as symbolic sequences (over event codes trigger) and as event frequencies over some aggregated temporal windows (lagged features). The previous data-driven PdM approach is to manually create rule-based predictive patterns for a particular component based on a Boolean combination of a few relevant event codes [16]. Such an approach is heavily experience-based and very time-consuming, but it illustrates an important concept that component failure can be predicted by checking daily logs for patterns. Many methods like Markov process [25–27], Bayesian network [36] were applied to enable PdM and CBM to estimate failure rate, deterioration of components, the breakdown of machines and effective scheduling to overcome problems. Our approach encapsulated the developed computational pipeline and it has obtained an optimal overall success to predict the ball bearing component.

Another major challenge in predictive maintenance is to collect faulty data for training the supervised model. From a machine learning perspective, the purpose is to collect normal state behavior to failure [37]. In real application, the failure event is not always easy to collect, thus leading to a high imbalanced setting. However, this data is important for learning a discriminative model between normal and failure samples by training a supervised machine learning model over two classes (normal state, anomaly state). In our approach, the target variable was expressed as RUL and estimated by calculating the remaining number of days to the failure event-based as provided by the internal service records. RUL variable has been discretized into a binary variable and tested for several windows period before machine down ($W = \{30, 20, 10\}$ days). This procedure allows setting the optimal RUL threshold in order to (i) accurate identify a high-risk of failure event and low-risk of failure event while (ii) alleviating the natural imbalanced setting of this task. In fact, the optimal binary RUL split was evaluated and tested into the classification step. Future work may be addressed to extend the binary formulation into a multi-class paradigm while alleviating the imbalanced setting.

The RUL prediction task is often affect by uncertainty. In the literature, uncertainty quantification in the RUL prediction is approached as an uncertainty propagation problem by using statistical models [38]. In our work, we deal with this problem by introducing a non-linear boosting methods to learn the intrinsic variability (e.g., variance) of representative data. However, the boosting methods are trained offline. We plan to extend this methodology as future work, by employing an online boosting training strategy that might be able to learn time-varying parameters in the presence of both non-linear and non-stationary conditions.

7. Conclusions

In conclusion, we presented a log-based PdM application by taking in advantage event-based triggers and the application of state-of-the-art machine learning techniques to build predictive models. The main advantage of such an approach is the use of aggregated event-based predictors (errors and warning events) as temporal characteristics to predict the potential machine down failures. The proposed application has been deployed a PdM application for woodworking by taking into advantage distributed a Big Data environment to generate data-driven predictive models that are based upon historical log data. The PdM application, deployed by a Big Data stream processing framework, screens log files and predicts the machine down the state of the woodworking machine every 24 h. Status of machine down failure is visualized through a front-end dashboard where the trend of predicted probability of each connected machine can be monitored in real-time. The presented approach applied to woodworking industrial machine could also apply to other domains, such as IT infrastructure, industrial or medical equipment in which equipment logs are recorded. Our model evaluation has shown a reliable predictive performance of the classifier (up to 98.9%, 99.6%, and 99.1% accuracy, recall, and precision respectively). Future work may be addressed to generalize the proposed system to other components or industrial machine types. In this context, domain-adaptation techniques can be exploited in order to mitigate the domain-shifts between training and test set. In this scenario, unsupervised domain adaptation methods can be exploited (e.g., [39]) by taking into account the difficulty to collect RUL information in specific industrial machine types. Another interesting future direction would be to model the temporal dynamics of each feature within pre-defined windows. To achieve this purpose the boosting based models could be extended to multiple instance boosting models.

Author Contributions: Conceptualization, M.C. (Matteo Calabrese), M.C. (Martin Cimmino), F.F., M.M. (Martina Manfrin), G.T., G.C., A.C., D.K.; methodology, M.C. (Matteo Calabrese), M.C. (Martin Cimmino), F.F., M.M. (Martina Manfrin), L.R., S.C., M.P., D.K.; software, M.C. (Matteo Calabrese), M.C. (Martin Cimmino), F.F., D.K.; validation, M.C. (Matteo Calabrese), M.C. (Martin Cimmino), F.F., D.K.; formal analysis, M.C. (Matteo Calabrese), M.C. (Martin Cimmino), F.F., M.M. (Martina Manfrin), L.R., S.C., M.P., D.K.; investigation, M.C. (Matteo Calabrese), M.C. (Martin Cimmino), F.F., M.M. (Martina Manfrin), L.R., S.C., M.P., M.M. (Maura Mengoni), E.F., D.K.; writing—original draft preparation L.R., S.C., M.P., D.K.; supervision G.T., G.C., A.C., M.M. (Maura Mengoni), E.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the BIESSE SpA and Accenture SpA.

Acknowledgments: We are very thankful to Marco Siciliano, Donato Concilio, Paccapeli Giancarlo, Giarratana Gianluca, Corasaniti Angela and Trasciatti Nito for their support and useful discussions on SOPHIA project.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. StdOut flag description.

StdOut Flag	Regular Expression	Format TXT	Description
START	START.*	START (18/10/2018 10:00:08)=BdriveIso/Step13 _strettaDX campo5 650x334x19 _20181018100008547,0,1,0,-1,1	Indicates the timestamp of program execution START
Start - BBOX_35	BBOX_35[0 – 9].1	PLC: 277545608678.051 BBOX_35(1 1)[6137216]	Indicates the start point of an execution center
BBOX	BBOX_.*	PLC: 15401013.151 BBOX_22(0 1)[4598]	All BBOX events
Errori PLC	ERRORE ?.*?=(PLC.*?) ?>.*	ERRORE (0-27/09/2018 08:47:25)=PLC 9314 >1/O device missing or incorrectly defined. ybn.10.0.10.PE75 (PE71-IN)	All PLC errors
End - BBOX_35	BBOX_35[0 – 9].0	PLC: 277545608678.051 BBOX_35(0 0)[6136089]	Indicate the end point of an execution center
ENDP	ENDP.*	ENDP (18/10/2018 09:59:57)=0,0	Indicates the timestamp of program execution END

Table A2. Error Classification of PLC error code. KO: unexpected stop error.

Error Classification Group	PLC Code
Emergency/inverter	PLC9004, PLC9705
Security control/inverter	PLC90455
KO	PLC9001, PLC9024, PLC9096, PLC9103, PLC9224, PLC9282, PLC9284, PLC9366, PLC9532, PLC9763, PLC9766, PLC9952, PLC9955, PLC90177, PLC90208, PLC90223, PLC90226, PLC90429, PLC90712, PLC90713, PLC90714, PLC90715, PLC90716, PLC90717, PLC90718, PLC91219, PLC91220
KO / Emergency	PLC9690
KO / Inverter	PLC9038
KO / Overheating	PLC9054, PLC9530, PLC9531, PLC90609
Emergency	PLC9005, PLC9035, PLC9068, PLC9191, PLC9448, PLC9517, PLC9643, PLC9689, PLC9954, PLC9996, PLC90062, PLC90467, PLC90473, PLC90607
Emergency/Security control	PLC90454, PLC90456, PLC90465, PLC90485, PLC90620, PLC90623, PLC90625, PLC90627, PLC90628
Inverter	PLC9029, PLC9093, PLC9514, PLC9644, PLC9746, PLC9044, PLC9915
Overheating	PLC90445, PLC90446, PLC90448, PLC90449, PLC9012, PLC90489, PLC90490
Overheating / inverter	PLC90500
Security control	PLC90457, PLC90458, PLC90459, PLC90461, PLC90469, PLC90470, PLC90477, PLC90478, PLC90481, PLC90482, PLC90483, PLC90621, PLC90622, PLC90624, PLC90626
Security control / inverter	PLC90462
Tool change	PLC9116, PLC9118, PLC9145, PLC9146, PLC9147, PLC9148, PLC9149, PLC9159, PLC9167, PLC9172, PLC9338, PLC9364, PLC9740, PLC9742

Table A3. Hyperparameter ranges and optimal values.

Model	Hyperparameter Ranges	Optimal Hyperparameters
Distributed Random Forest (DRF)	Max depth = [5, 6, 7, 8, 9] Number of trees = [10, 20, 30, 40, 50] Sample rate = [0.5, 0.6, 0.7, 0.8, 1.0] Column sample rate per tree = [0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 1.0]	Max depth = 9 Number of trees = 30 Sample rate = 0.8 Column sample rate per tree = 0.5
Gradient Boosting Machine (GBM)	Learning rate = [0.05, 0.1, 0.2] Max depth = [5, 6, 7, 8, 9] Number of trees = [50, 100, 150] Sample rate = [0.7, 0.8, 0.9]	Learning rate = 0.2 Max depth = 9 Number of trees = 150 Sample rate = 0.7
Extreeme Gradient Boosting (XGBoost)	Learning rate = [0.05, 0.1, 0.2] Max depth = [4, 5, 6, 7, 8, 9] Number of trees = [50, 100, 150, 200, 250, 300] Sample rate = [0.7, 0.8, 0.9] Column sample rate per tree = [0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 1.0]	Learning rate = 0.05 Max depth = 9 Number of trees = 300 Sample rate = 0.7 Column sample rate per tree = 0.7

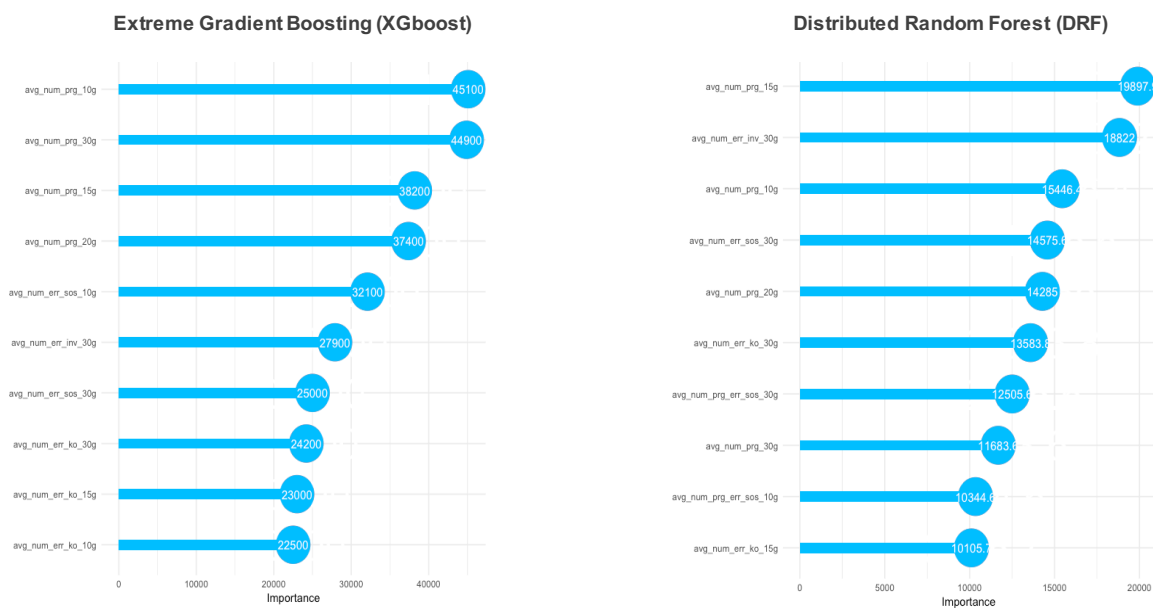


Figure A1. Variable importance of XGboost (left side) and DRF classifiers (right side). Horizontal plot shows the top 10 variables selected of the model to classify machine down against control group.

References

1. Mönch, L.; Fowler, J.W.; Dauzere-Peres, S.; Mason, S.J.; Rose, O. A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *J. Sched.* **2011**, *14*, 583–599. [[CrossRef](#)]
2. Susto, G.A.; Schirru, A.; Pampuri, S.; Pagano, D.; McLoone, S.; Beghi, A. A predictive maintenance system for integral type faults based on support vector machines: An application to ion implantation. In Proceedings of the 2013 IEEE International Conference on Automation Science and Engineering (CASE), Madison, WI, USA, 17–21 August 2013; pp. 195–200.
3. Paolanti, M.; Romeo, L.; Felicetti, A.; Mancini, A.; Frontoni, E.; Loncarski, J. Machine learning approach for predictive maintenance in industry 4.0. In Proceedings of the 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Oulu, Finland, 2–4 July 2018; pp. 1–6.
4. Susto, G.A.; Pampuri, S.; Schirru, A.; Beghi, A.; De Nicolao, G. Multi-step virtual metrology for semiconductor manufacturing: A multilevel and regularization methods-based approach. *Comput. Oper. Res.* **2015**, *53*, 328–337. [[CrossRef](#)]
5. Romeo, L.; Loncarski, J.; Paolanti, M.; Bocchini, G.; Mancini, A.; Frontoni, E. Machine learning-based design support system for the prediction of heterogeneous machine parameters in industry 4.0. *Expert Syst. Appl.* **2020**, *140*, 112869. [[CrossRef](#)]
6. Romeo, L.; Paolanti, M.; Bocchini, G.; Loncarski, J.; Frontoni, E. An Innovative Design Support System for Industry 4.0 Based on Machine Learning Approaches. In Proceedings of the 2018 5th International Symposium on Environment-Friendly Energies and Applications (EFEA), Rome, Italy, 24–26 September 2018; pp. 1–6.
7. Susto, G.A.; Beghi, A.; De Luca, C. A predictive maintenance system for epitaxy processes based on filtering and prediction techniques. *IEEE Trans. Semicond. Manuf.* **2012**, *25*, 638–649. [[CrossRef](#)]
8. Coleman, C.; Coleman, C.; Damodaran, S.; Chandramouli, M.; Deuel, E. *Making Maintenance Smarter: Predictive Maintenance and the Digital Supply Network*; Deloitte University Press: New York, NY, USA, 2017.
9. Fraser, K.; Hvolby, H.H.; Watanabe, C. A review of the three most popular maintenance systems: How well is the energy sector represented? *Int. J. Glob. Energy Issues* **2011**, *35*, 287–309. [[CrossRef](#)]
10. Chen, F. Issues in the continuous improvement process for preventive maintenance: Observations from Honda, Nippondenso and Toyota. *Prod. Inventory Manag. J.* **1997**, *38*, 13–16.

11. Lei, Y.; Li, N.; Gontarz, S.; Lin, J.; Radkowski, S.; Dybala, J. A model-based method for remaining useful life prediction of machinery. *IEEE Trans. Reliab.* **2016**, *65*, 1314–1326. [[CrossRef](#)]
12. Yoon, J.T.; Youn, B.D.; Yoo, M.; Kim, Y.; Kim, S. Life-cycle maintenance cost analysis framework considering time-dependent false and missed alarms for fault diagnosis. *Reliab. Eng. Syst. Saf.* **2019**, *184*, 181–192. [[CrossRef](#)]
13. Sakib, N.; Wuest, T. Challenges and Opportunities of Condition-based Predictive Maintenance: A Review. *Procedia CIRP* **2018**, *78*, 267–272. [[CrossRef](#)]
14. Lei, Y.; Li, N.; Guo, L.; Li, N.; Yan, T.; Lin, J. Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mech. Syst. Signal Process.* **2018**, *104*, 799–834. [[CrossRef](#)]
15. Sipos, R.; Fradkin, D.; Moerchen, F.; Wang, Z. Log-based predictive maintenance. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 1867–1876.
16. Wang, J.; Li, C.; Han, S.; Sarkar, S.; Zhou, X. Predictive maintenance based on event-log analysis: A case study. *IBM J. Res. Dev.* **2017**, *61*, 11–121. [[CrossRef](#)]
17. Jardine, A.K.; Lin, D.; Banjevic, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mech. Syst. Signal Process.* **2006**, *20*, 1483–1510. [[CrossRef](#)]
18. Heng, A.; Tan, A.C.; Mathew, J.; Montgomery, N.; Banjevic, D.; Jardine, A.K. Intelligent condition-based prediction of machinery reliability. *Mech. Syst. Signal Process.* **2009**, *23*, 1600–1614. [[CrossRef](#)]
19. Khan, S.; Yairi, T. A review on the application of deep learning in system health management. *Mech. Syst. Signal Process.* **2018**, *107*, 241–265. [[CrossRef](#)]
20. Xia, T.; Dong, Y.; Xiao, L.; Du, S.; Pan, E.; Xi, L. Recent advances in prognostics and health management for advanced manufacturing paradigms. *Reliab. Eng. Syst. Saf.* **2018**, *178*, 255–268. [[CrossRef](#)]
21. Goldszmidt, M. Bayesian network classifiers. In *Wiley Encyclopedia of Operations Research and Management Science*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2010.
22. Gullà, F.; Cavalieri, L.; Ceccacci, S.; Germani, M. A BBN-based Method to Manage Adaptive Behavior of a Smart User Interface. *Procedia CIRP* **2016**, *50*, 535–540. [[CrossRef](#)]
23. Ghahramani, Z. Learning dynamic Bayesian networks. In *International School on Neural Networks, Initiated by IIASS and EMFCSC*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 168–197.
24. Arroyo-Figueroa, G.; Sucar, L.E. A temporal Bayesian network for diagnosis and prediction. *arXiv* **2013**, arXiv:1301.6675.
25. Salfner, F. Predicting Failures with Hidden Markov Models. In Proceedings of the 5th European Dependable Computing Conference (EDCC-5), Budapest, Hungary, 20–22 April 2005.
26. Vriagnet, P.; Avila, M.; Duculty, F.; Kratz, F. Failure event prediction using hidden markov model approaches. *IEEE Trans. Reliab.* **2015**, *64*, 1038–1048. [[CrossRef](#)]
27. Zhou, Z.J.; Hu, C.H.; Xu, D.L.; Chen, M.Y.; Zhou, D.H. A model for real-time failure prognosis based on hidden Markov model and belief rule base. *Eur. J. Oper. Res.* **2010**, *207*, 269–283. [[CrossRef](#)]
28. Canizo, M.; Onieva, E.; Conde, A.; Charramendieta, S.; Trujillo, S. Real-time predictive maintenance for wind turbines using Big Data frameworks. In Proceedings of the 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), Dallas, TX, USA, 19–21 June 2017; pp. 70–77.
29. Bousdekis, A.; Hribernik, K.; Lewandowski, M.; von Stietencron, M.; Thoben, K.D. A Unified Architecture for Proactive Maintenance in Manufacturing Enterprises. In *Enterprise Interoperability VIII*; Springer: Cham, Switzerland, 2019; pp. 307–317. [[CrossRef](#)]
30. Liu, Z.; Jin, C.; Jin, W.; Lee, J.; Zhang, Z.; Peng, C.; Xu, G. Industrial AI Enabled Prognostics for High-speed Railway Systems. In Proceedings of the 2018 IEEE International Conference on Prognostics and Health Management (ICPHM), Seattle, WA, USA, 11–13 June 2018; pp. 1–8. [[CrossRef](#)]
31. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
32. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [[CrossRef](#)]
33. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
34. Lipton, Z.C. The mythos of model interpretability. *Queue* **2018**, *16*, 31–57. [[CrossRef](#)]

35. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [[CrossRef](#)]
36. Abu-Samah, A.; Shahzad, M.; Zamai, E.; Said, A.B. Failure prediction methodology for improved proactive maintenance using Bayesian approach. *IFAC-PapersOnLine* **2015**, *48*, 844–851. [[CrossRef](#)]
37. Xu, Y.; Sun, Y.; Liu, X.; Zheng, Y. A digital-twin-assisted fault diagnosis using deep transfer learning. *IEEE Access* **2019**, *7*, 19990–19999. [[CrossRef](#)]
38. Gan, C.L. *Prognostics and Health Management of Electronics: Fundamentals, Machine Learning, and the Internet of Things*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2020.
39. Sun, B.; Feng, J.; Saenko, K. Return of frustratingly easy domain adaptation. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).